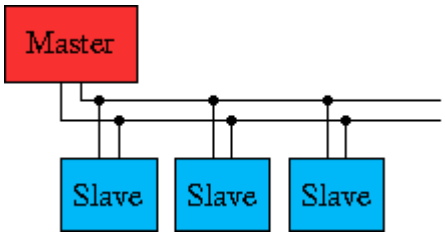
I2C - Bus

1.Eigenschaften

Der Name I2C - Bus ist eine Abkürzung für Inter IC Bus

Der I2C Bus ist als serielle Verbindung zwischen einzelnen elektronischen Komponenten innerhalb eines Gerätes konzipiert. Der I2C-Bus wurde bereits vor etwa 20 Jahren von Philips entwickelt. Ziel war ein hierarchisches Bussystem (Master-Slave Architektur) für geringe Datenraten, an das sich bei geringstem Verdrahtungsaufwand mehrere IC's anschließen lassen. Die maximale Anzahl der Slaves beträgt 112.



Damit bei diesen IC's auf teure Komponenten wie Quarze oder Takterzeuger verzichtet werden konnte, sollte der Bus den Takt übertragen. Durch die Normierung wird der Aufbau und die Konzeption der Software vereinfacht. Es werden zahlreiche Komponenten mit I2C Bus – Interface unterstützt.

Beispiele : Speicher, Uhren, Tuner, Mikrocontroller, Audio-Chips, Temperatursensoren, Drehzahlmesser, A/D und D/A Wandler

Es gibt kaum ein Fernsehgerät oder einen Videorekorder aus europäischer Fertigung, in welchem nicht der I^2C-Bus zur Steuerung eingesetzt wird. Man findet ihn in den HiFi-Anlagen im Heim, in Autoradios und manchmal sogar in Telefonen und Kassenterminals. Pauschal lässt sich sagen: wenn ein Gerät der Unterhaltungselektronik zur Steuerung seiner Funktion einen Mikroprozessor benötigt, so erfolgt die Steuerung in den meisten Fällen über den seriellen I^2C-Bus.

Geschwindigkeiten :

Standard Mode:	100 kBit / s
Fast Mode:	400 kBit / s
High Speed Mode:	3.5 MBit / s

Der Bus ist Multi – Master fähig. Es können mehrere Master auf einem Bus angeschlossen werden, die sich mit speziellen Arbitrations Methoden den alleinigen Zugriff auf den Bus sichern und Kollisionen feststellen.

2. Hardware

2.1. Verdrahtung

Der I2C-Bus basiert auf zwei Leitungen:

SCL transportiert das Taktsignal. Dieses wird ausschließlich vom Busmaster erzeugt und von den angeschlossenen Chips gelesen, die Übertragungsrichtung ist daher eindeutig festgelegt.

SDA überträgt die Datenbits. Über diesen Anschluss senden sowohl Master als auch Slave ihre Daten. Der oder die empfangenden Chips müssen dabei SDA jeweils auf High-Potential legen.

Hardware Beschaltung der Leitungen SDA und SCL :

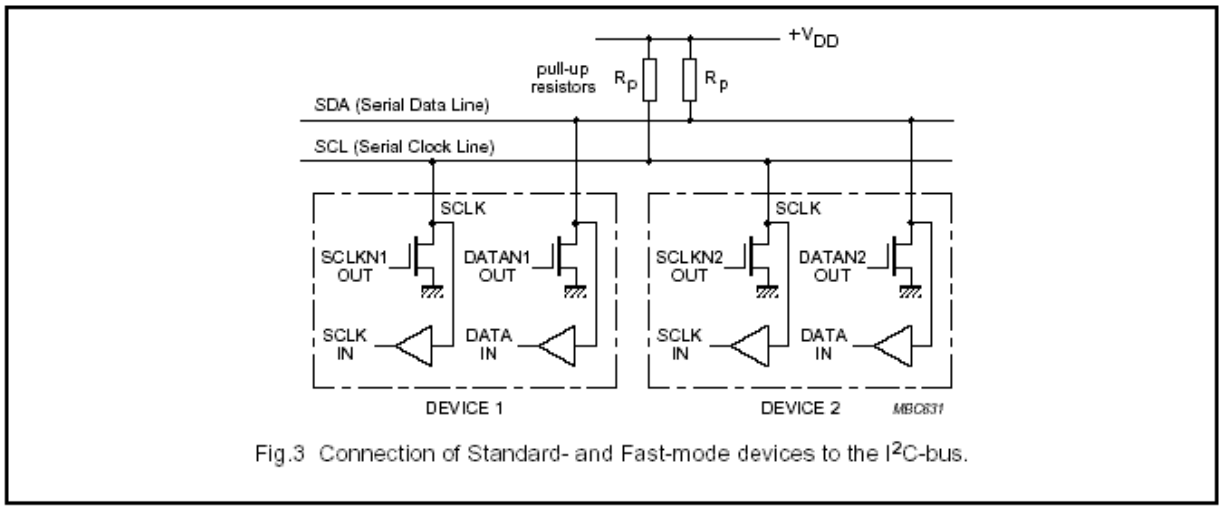


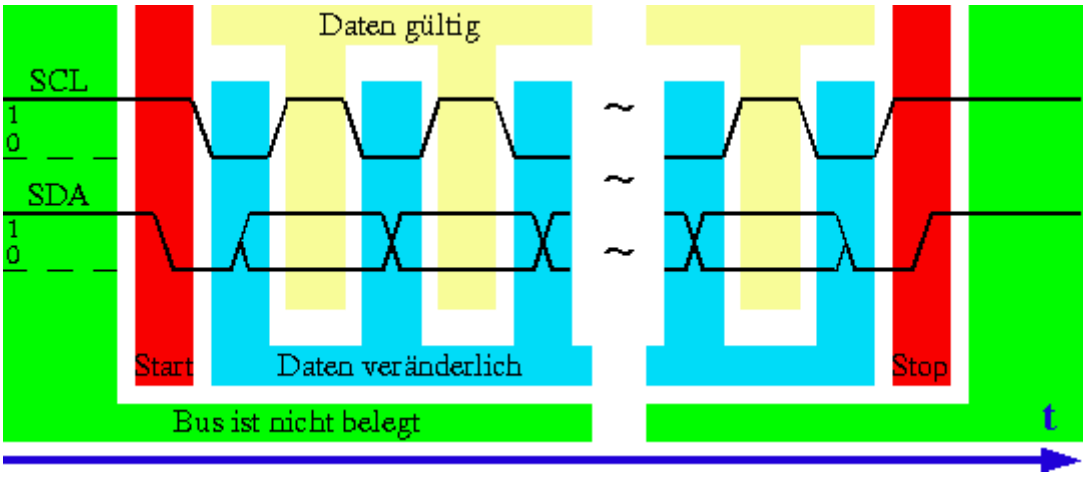
Fig.3 Connection of Standard- and Fast-mode devices to the I²C-bus.

Die aktuellen Logik – Level hängen von VDD ab und sind auf 0.3 VDD Input Low und 0.7 VDD Input High näherungsweise festgelegt. Die Anzahl der Bausteine am Bus hängt von der gesamten kapazitiven Belastung ab, die 400 pF nicht übersteigen darf.

2.2. Buszustände

Der Zustand des auf SDA angelegten Datenbits darf nur wechseln, während SCL=0 ist. Bei SCL=1 sind die angelegten Daten grundsätzlich gültig. Dies ist unabhängig davon, ob der Master sendet und der Slave empfängt, oder umgekehrt.

Die Ausnahme von der Regel bilden spezielle Steuerkommandos, die auf einem Wechsel auf der Datenleitung während SCL=1 basieren und immer vom Master gesendet werden.



Im Grundzustand sind SCL und SDA high.

Will der Master eine Abfrage an ein Device senden, so beginnt er immer mit einer **START-Condition**. Dabei wird SDA von High nach Low gesetzt, während SCL High ist.

Anschließend wird SCL auf Low gesetzt. Solange SCL Low ist, darf sich der Zustand von SDA ändern. Daher kann in diesem Zustand der Sender ein Datenbit auf den Bus geben, welches vom Empfänger interpretiert wird, sobald SCL vom Master wieder auf High gesetzt wird. Daten werden grundsätzlich immer mit dem höchstwertigen Bit zuerst übermittelt.

Nachdem 8 Bits auf diese Weise (SCL low, Datenbit auf SDA, SCL high, warten/empfangen, SCL low) übertragen wurden, folgt die Bestätigung vom Empfänger, dass diese Daten korrekt empfangen wurden. Diese **ACK-Condition** ist nichts weiter als ein weiteres Bit, nur mit vertauschten Rollen: der Empfänger der 8 übermittelten Bits sendet im 9. Bit seinerseits eine 0, um den Empfang zu quittieren. Daher muss der Sender als 9. Bit einen High-Zustand auf SDA ausgeben und prüfen, ob dieser vom Empfänger auf Low gezogen wird. Stellt der Sender fest, dass kein ACK als Bestätigung erfolgt, so hat der Empfänger vielleicht einen Taktpuls verpasst, wurde zu schnell mit Daten überflutet oder ist schlicht kaputt. Die beste Verhaltensweise bei fehlendem ACK ist es, sofort eine STOP-Condition auszugeben und die Übertragung noch einmal zu versuchen.

Das Ende einer I2C-Übertragung wird durch eine **STOP-Condition** angezeigt. Dafür setzt der Master - nur er darf eine Übertragung starten und beenden - SDA von Low nach High, solange SCL high ist.

Abschließend ist der Bus wieder im Grundzustand: SCL und SDA sind high.

Braucht ein Slave noch Zeit zur Verarbeitung kann der Slave die SCL Leitung auf Low halten, dadurch weiß der Master, dass er mit der Übertragung des nächsten Bytes warten muß.

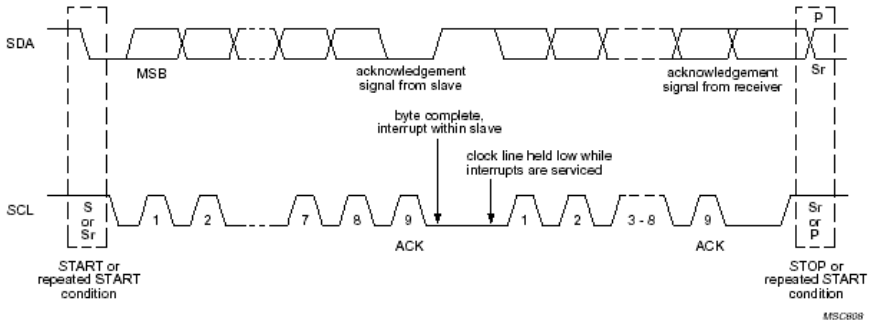


Fig.6 Data transfer on the I²C-bus.

2.3. Adressierung

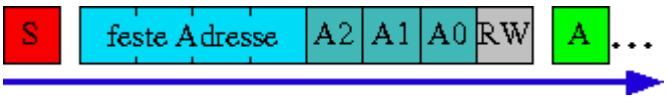
Wie Eingangs erwähnt, ist der I2C-Bus für den Anschluss vieler I2C-Slaves konzipiert. Damit dies reibungslos funktioniert, muss jeder I2C-Chip genau wissen, welche Daten für ihn bestimmt sind und welche einem anderen Empfänger zugedacht sind.

Dazu wird folgender Mechanismus genutzt: Jeder I2C-Chiptyp besitzt eine im Chip hart verdrahtete 4 Bit-Adresse. Desweiteren hat jeder I2C-Chip bis zu 3 Adresspins, die mit einer beliebigen Adresse belegt werden können. Damit lassen sich an jeden I2C-Bus bis zu 8 Chips mit gleicher Deviceadresse bei 16 möglichen verschiedenen Deviceadressen anschließen und unterscheiden, beispielsweise 8

Busextender, 8 EEPROM's und 8 LED-Treiberbausteine.

Allerdings haben nicht alle Chips wirklich nach aussen geführte Adresspins. Es gibt auch einige EEPROM's mit intern fest verschalteten Adressen, obwohl die Adresspins nach aussen geführt sind - wahrscheinlich um Pinkompatibitlität zu den EEPROM's anderer Hersteller herbeizuführen. Ebenfalls zu bekommen sind ansonsten baugleiche Chips mit verschiedenen fest einprogrammierten Adressen. Beispielsweise sind die I/O-Expander PCF8574 und PCF8574A völlig identisch - bis auf eine unterschiedliche Deviceadresse. So lassen sich an einem Bus 16 I/O-Expander anschließen und nicht nur 8. Daher ist ein Blick in das genaue Datenblatt des Chips unbedingt anzuraten, Datenblätter von baugleichen Typen unterschiedlicher Hersteller genügen nicht. Und so wird der Adressierungsmechanismus beim I2C-Bus eingesetzt:

Unmittelbar nach dem Senden der **Start-Condition** wird ein Steuerbyte mit einem weiter unten beschriebenen Aufbau übertragen, das von allen am Bus lauschenden Slaves ausgewertet wird.
Die 4 höchstwertigen und daher zuerst gesendeten Bits enthalten die fest eingebrannte Device-Adresse.



Adressen gebräuchlicher Chips könnten aus ihren Datenblättern entnommen werden.
Daraufhin folgen drei Adressbits, die den Adresspins am Chip entsprechen. Besitzt der Chip weniger als drei Adresspins, so sind die überzähligen Bits in der Regel 0.

Das letzte und niederwertigste Bit muss eine 1 für nun folgende Leseoperationen bzw. eine 0 für Schreiboperationen enthalten.
Nun kommt die nach jedem gesendeten Byte fällige **ACK-Condition**. Alle am Bus hängenden I2C-Devices wissen jetzt, ob die bis zur den Buszyklus abschließenden **Stop-Condition** gesendeten Daten für sie bestimmt sind oder nicht.

Reservierte Adressen:

Table 2 Definition of bits in the first byte

SLAVE ADDRESS	$\overline{R/W}$ BIT	DESCRIPTION
0000 000	0	General call address
0000 000	1	START byte(1)
0000 001	X	CBUS address(2)
0000 010	X	Reserved for different bus format(3)
0000 011	X	Reserved for future purposes
0000 1XX	X	Hs-mode master code
1111 1XX	X	Reserved for future purposes
1111 0XX	X	10-bit slave addressing

Da mit der 7-Bit Adressierung nur 112 verschiedene Adressen möglich sind, wurde in der I2C-Spezi kation 1.0 von 1992 die 10-Bit Adressierung eingeführt. Dadurch sind 1024 weitere Adressen verfügbar. Die 10-Bit Adressierung funktioniert dadurch, dass nach der START-Condition erst eine spezielle 7-Bit Adresse (11110XX) mit einem Schreibbit (0) übertragen wird, wobei die beiden niedrigstwertigen Bits dieser 7-Bit Adresse die beiden höchstwertigen Bits der 10-Bit Adresse sind. Weil die 7-Bit Adresse zu den reservierten Adressen gehört, wird kein Gerät darauf antworten, das nur 7-Bit Adressierung versteht, sondern
ausschlieÿlich Geräte, die eine 10-Bit Adresse haben, deren beiden höchstwertigen Bits den beiden niederwertigsten Bits der 7-Bit Adresse entsprechen. Nachdem ein Slave auf die Adresse geantwortet hat, kann nun der Master ein weiteres Datenbyte übertragen, das nun die nächsten 8 Bit der Adresse enthält. Wenn auf diese Adresse ein Slave mit einem ACK antwortet, so kann der Master anfangen
Daten zu senden.
Sollte der Master allerdings Daten vom Slave lesen, so muss er eine wiederholte START-Condition auslösen und wieder die 7-Bit Adresse senden, allerdings diesmal mit einem Lesebit (11110XX1). Dadurch erkennt der zuvor adressierte Slave, dass er als Slave-Transmitter funktionieren soll und fängt an, Daten zu senden.

2.4. Datenübertragung

Jedes übertragene Byte ist 8 Bit lang und wird mit dem MSB – Bit als erstes gesendet. Die Anzahl der Bytes ist unbeschränkt. Kann der Empfänger die empfangenen Daten nicht schnell genug weiterverarbeiten, so kann er die SCL Leitung auf Low halten, um dem Sender bei der Datenübertragung anzuhalten. Die Datenübertragung wird wieder aufgenommen wenn der Empfänger die SCL – Leitung wieder frei gibt (auf High setzt). Wird vom Slave kein ACK gesendet so erzeugt der Master eine Stop – Condition und bricht die Übertragung ab oder er startet mit einer Start Condition eine neue Übertragung.

2.5. Datenempfang des Masters

Werden Daten vom Slave zum Master übertragen, so sendet der Master die Adresse mit gesetztem R/W Bit. Die nachfolgenden Bytes werden vom Slave auf den Bus gegeben. Will der Master die Übertragung beenden, so wird das letzte Byte nicht mit der ACK-Condition bestätigt. Der Slave muss daraufhin die Datenleitung freigeben, damit der Master die Stop-Condition erzeugen kann.

2.6. Multimaster Betrieb

Wenn mehrere Master auf dem Bus arbeiten, muss ein Arbitrationsprozess ein Chaos vermeiden. Beginnen 2 Master gleichzeitig zu senden, so muss jeder Master den Zustand der SDA – Leitung kontrollieren. Sendet ein Master eine 1 und detektiert aber eine 0, so muss er seine Aktivitäten am Bus einstellen. Der Master mit der niedrigsten Adresse gewinnt. Der Clock wird aufgrund der wired-And Verknüpfung der einzelnen Bausteine synchronisiert.

2.7. Häufige Bausteine mit I2C Anschluss

Chip	Beschreibung	Device-Adresse	prog. Adressen
PCF8573	Uhrenbaustein mit Alarm und Spannungswächter	1101	4
PCF8583	Uhrenbaustein mit 240 Byte RAM	1010	2
PCF8570	256 Byte Low Voltage-SRAM	1010	8
24C01 bis 24C512	Serielle EEPROM's Die Nummer hinter dem 'C' entspricht der Kapazität in Kilobit	1010	0 bis 8
PCF8575	16 Bit I/O-Erweiterung	0100	8
PCF8574	8 Bit I/O-Erweiterung	0100	8
PCF8574A	8 Bit I/O-Erweiterung	0111	8
SAA1064	LED-Treiber für 4x8 LED's bzw. 4 x 7seg. Anzeigen mit Dezimalpunkt.	0111	2
PCF8591	AD-Wandler	1001	8

2.8. Echtzeituhr PCF8583

2.8.1. Eigenschaften:

240x8 bit RAM (Batteriebetrieb geeignet, 1 – 6V)

Stromaufnahme: 50uA

4 Jahres-Kalender, 12 oder 24 Stunden Format

I2C Interface für Jahr, Monate, Wochentag, Stunden, Minuten, Sekunden, Sekunden/100

Alarmregister

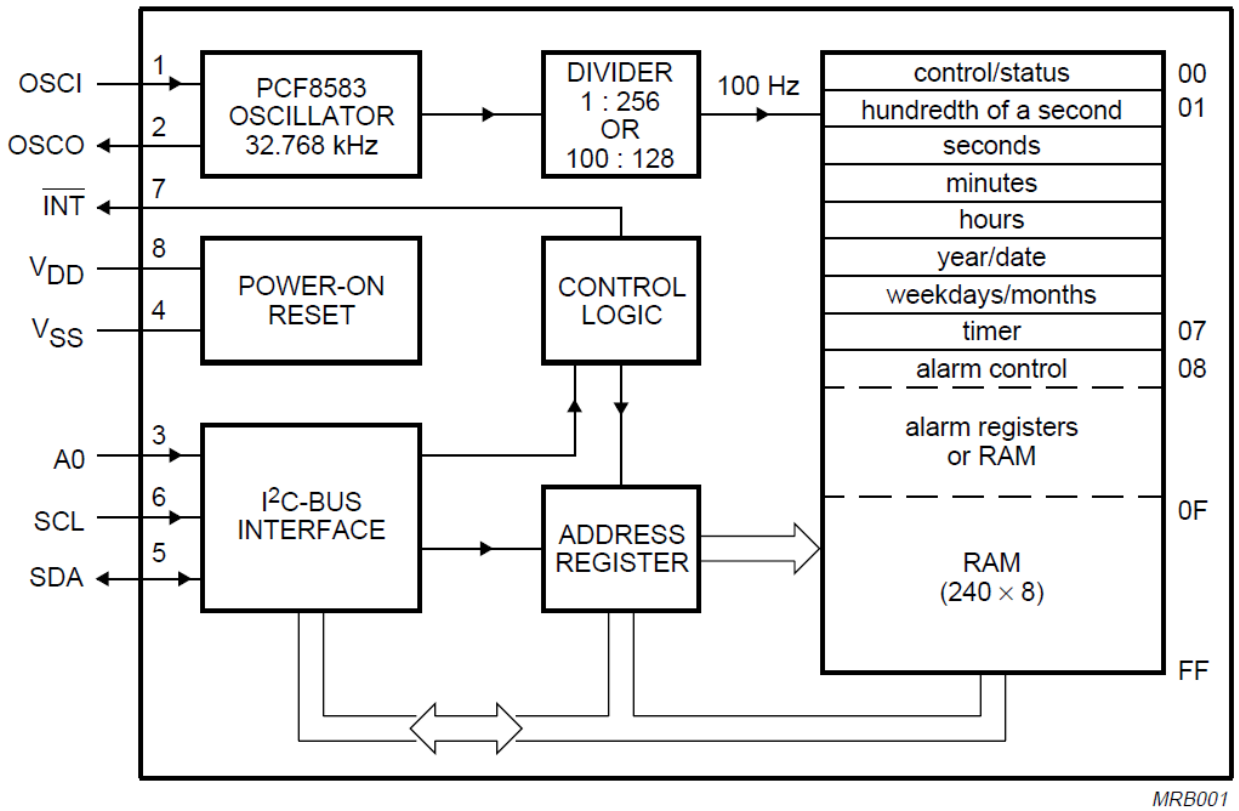
On-chip power fail detector

Battery backup

Quarz mit 32.768 kHz

2.8.2. Beschreibung

Aufbbau:



SYMBOL	PIN	DESCRIPTION
OSCI	1	oscillator input, 50 Hz or event-pulse input
OSCO	2	oscillator output
A0	3	address input
V_{SS}	4	negative supply
SDA	5	serial data line
SCL	6	serial clock line
$\overline{INT}$	7	open drain interrupt output (active LOW)
V_{DD}	8	positive supply

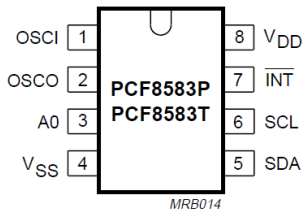


Fig.2 Pinning diagram.

Der PCF8583 enhält ein 256Byte RAM mit einem autoincrement Register. Die ersten 16 Byte sind special function Register zur Steuerung des Bausteines und zum Auslesen der aktuellen Uhrzeit. Die Adresse 00 ist das Status und Control Register. Adressen 1 bis 7 sind die Zählregister für die Uhrzeit. Adressen 8 bis 15 sind Alarmregister. Ein Timerregister kann für Timer-Events verwendet werden, bei einem Overflow wird ein Ausgang auf 0 gesetzt.

Oszillator :

Der Oszillator schwingt mit 32.768 Hz und kann mit einem Kondensator gegen VDD abgestimmt werden. Die Oszillatorfrequenz wird auf 100Hz geteilt. Es gibt auch einen 50Hz Mode, um die Zeitbasis aus der Netzfrequenz zu erhalten.

Setzen der Zeit:

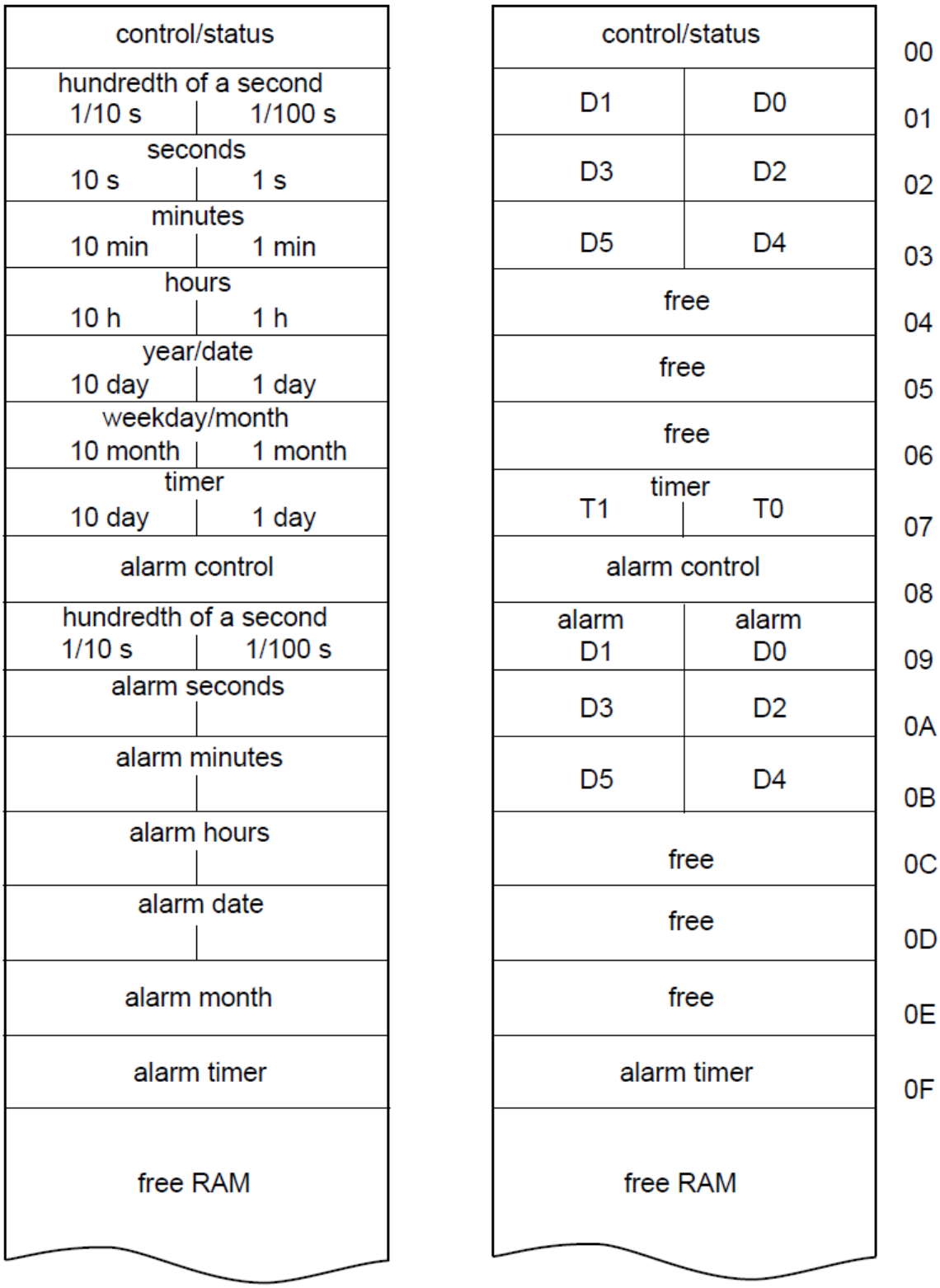
Die Uhr soll angehalten werden, um keine Fehlfunktion der Uhr zu provozieren.

Alarmregister:

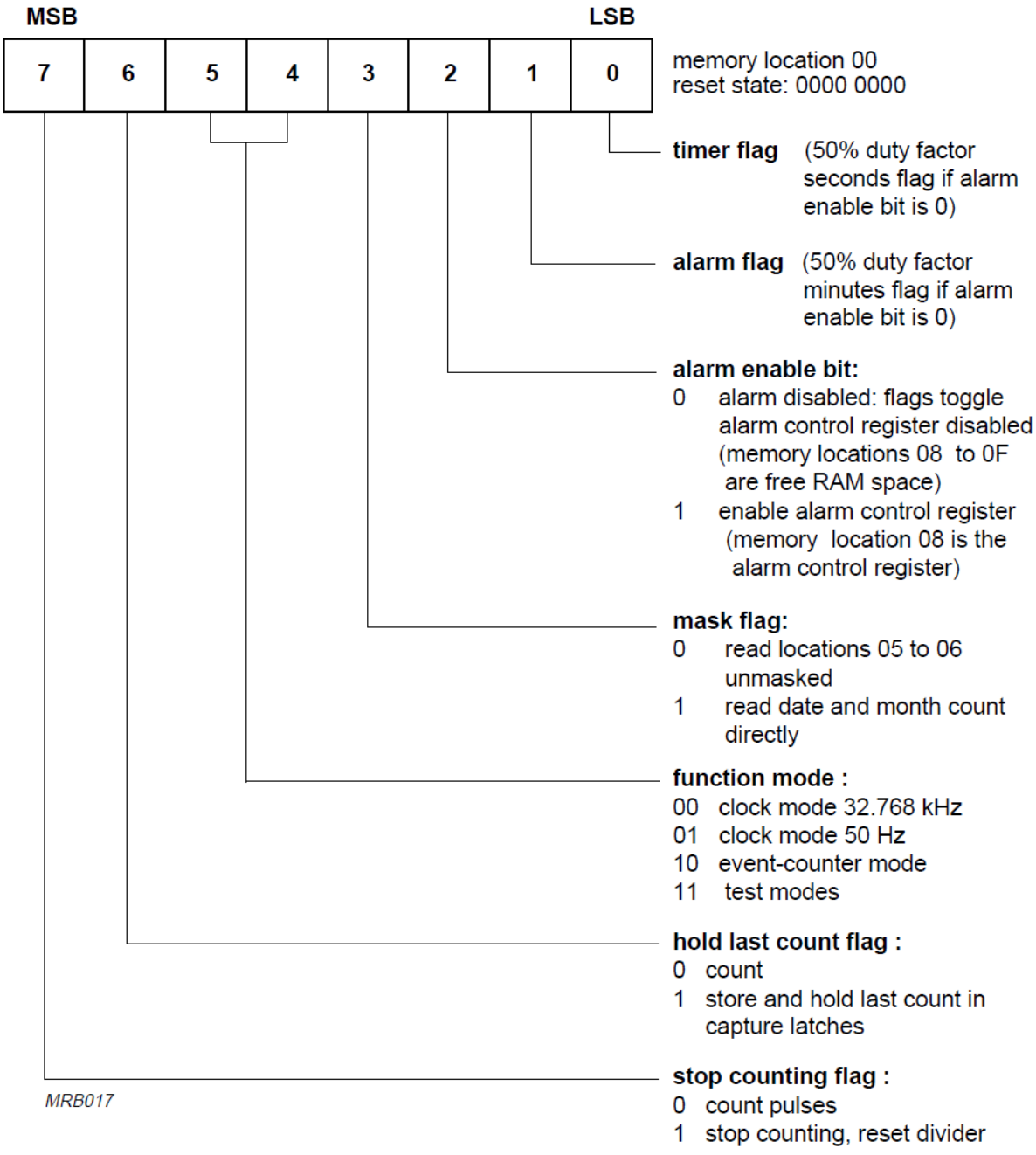
Das Alarmregister wird mit der aktuellen Zeit durch den Komparator verglichen. Bei Übereinstimmung wird ein Signal ausgegeben, der Zustand kann über den I2C Bus ausgelesen werden.

Verwendung : z.B: Aufwecken eines Prozssors aus einem Power-Down Modus.

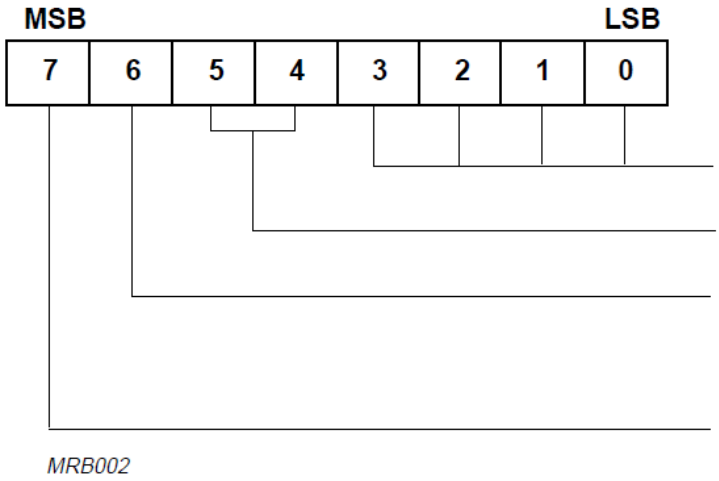
2.9. Register / RAM



Control Register:



Stundenzähler



memory location 04 (hours counter)
reset state: 0000 0000

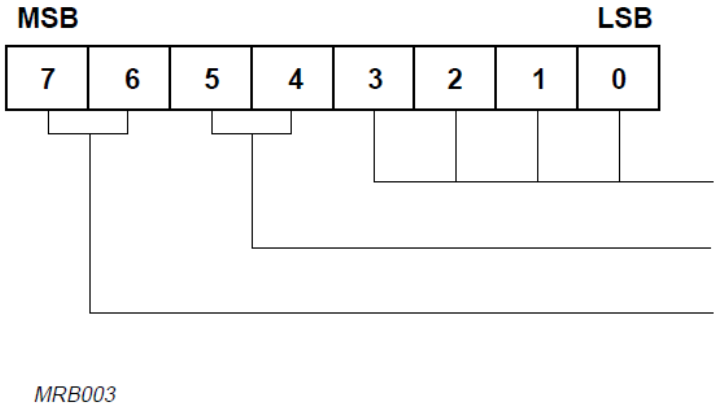
unit hours BCD

ten hours (0 to 2 binary)

AM/PM flag:
0 AM
1 PM

format:
0 24 h format, AM/PM flag remains unchanged
1 12 h format, AM/PM flag will be updated

Jahr:



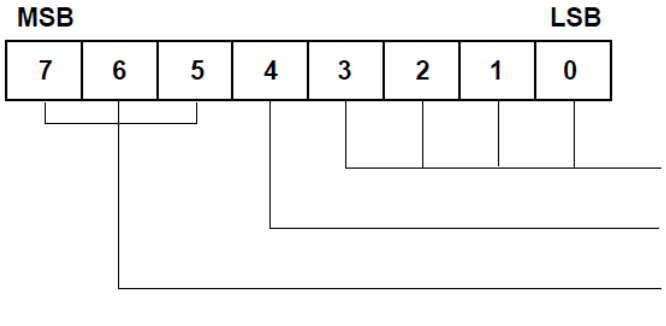
memory location 05 (year/date)
reset state: 0000 0001

unit days BCD

ten days (0 to 3 binary)

year (0 to 3 binary, read as 0 if the mask flag is set)

Wochentage / Monate



memory location 06 (weekdays/months)
reset state: 0000 0001

unit months BCD

ten months

weekdays (0 to 6 binary, read as 0 if the mask flag is set)

Zählerwerte – Überlauf:

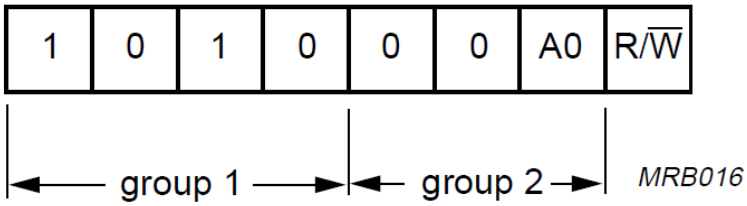
UNIT	COUNTING CYCLE	CARRY TO NEXT UNIT	CONTENTS OF THE MONTH COUNTER
Hundredths of a second	00 to 99	99 to 00	–
Seconds	00 to 59	59 to 00	–
Minutes	00 to 59	59 to 00	–
Hours (24 h)	00 to 23	23 to 00	–
Hours (12 h)	12 AM	–	–
	01 AM to 11 AM	–	–
	12 PM	–	–
	01 PM to 11 PM	11 PM to 12 AM	–
Date	01 to 31	31 to 01	1, 3, 5, 7, 8, 10 and 12
	01 to 30	30 to 01	4, 6, 9 and 11
	01 to 29	29 to 01	2, year = 0
	01 to 28	28 to 01	2, year = 1, 2 and 3
Months	01 to 12	12 to 01	–
Year	0 to 3	–	–
Weekdays	0 to 6	6 to 0	–
Timer	00 to 99	no carry	–

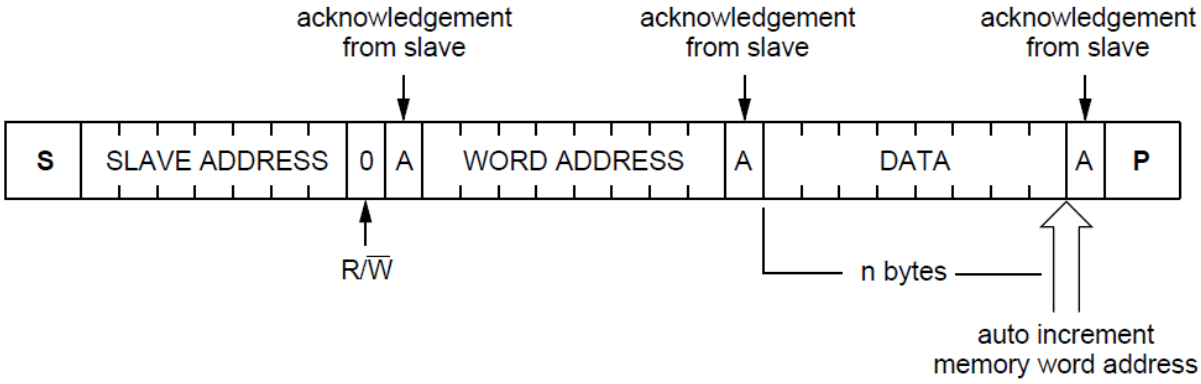
2.10.I2C – Bus Protokoll

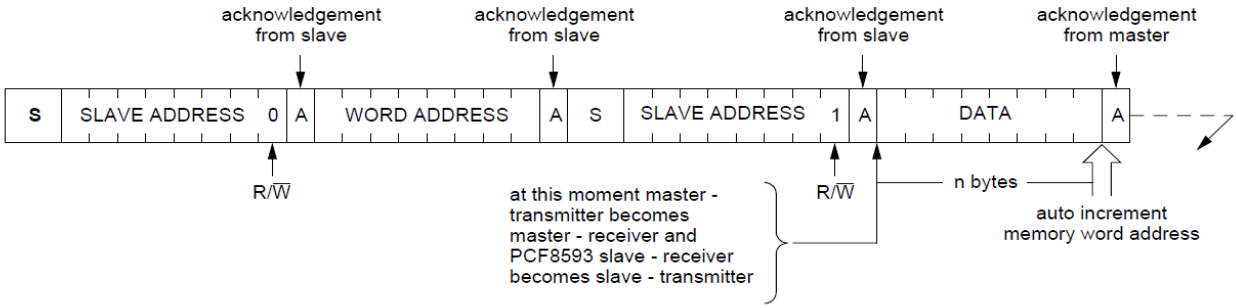
Der Baustein kann über den I2C – Bus angesprochen werden.

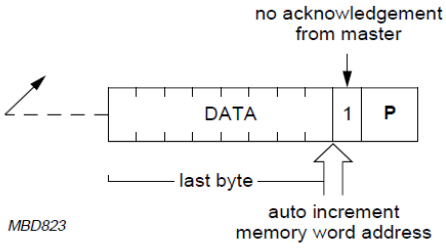
Der Uhrenbaustein ist ein Slave, daher ist der SCL Anschluss nur ein Eingang.

Als erstes muss immer die Adresse 101000xB (0x68 – 0x6B) gesendet werden. Die untersten 2 Bit werden durch die Anschlüsse A0 und A1 bestimmt.

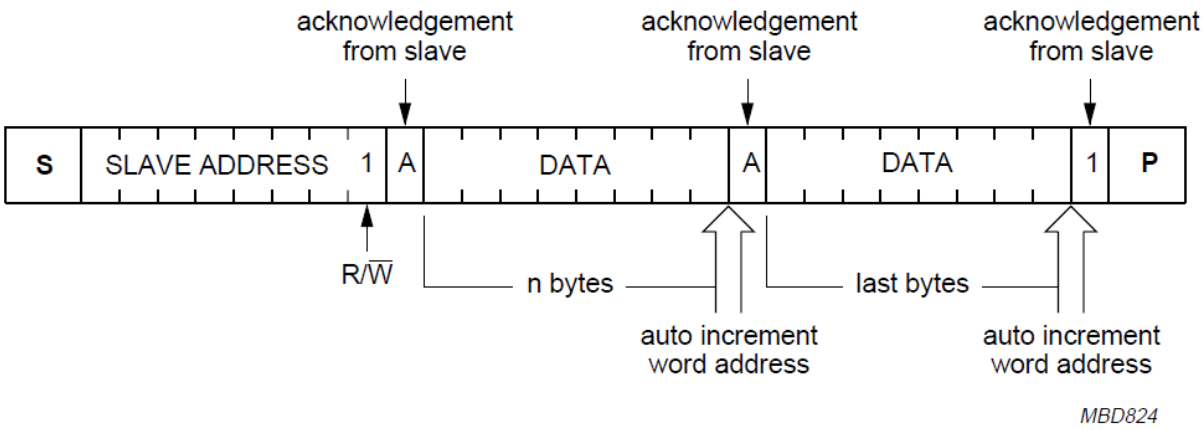








Der Master liest unmittelbar nach dem Schreiben der Slave-Adresse, ohne eine Stopbedingung zu generieren.



Master liest sofort, ohne die Adresse zu setzen (Read Mode).