

I2C-Bus - Inter-Integrated Circuit bus

Public Member Functions

```
#include <I2C.h>
```

	I2C (PinName sda, PinName scl) Create an I2C Master interface, connected to the specified pins.
void	frequency (int hz) Set the frequency of the I2C interface.
int	read (int address, char *data, int length, bool repeated=false) Read from an I2C slave.
int	read (int ack) Read a single byte from the I2C bus.
void	write (int address, const char *data, int length, bool repeated=false) Write to an I2C slave.
int	write (int data) Write single byte out on the I2C bus.
void	start (void) Creates a start condition on the I2C bus.
void	stop (void) Creates a stop condition on the I2C bus.
int	transfer (int address, const char *tx_buffer, int tx_length, char *rx_buffer, int rx_length, const event_callback_t &callback, int event=I2C_EVENT_TRANSFER_COMPLETE, bool repeated=false) Start non-blocking I2C transfer.
void	abort_transfer () Abort the on-going I2C transfer.

Zum Zugriff auf den I2C-Bus werden üblicherweise für den Schreibvorgang und für den Lesevorgang Arrays mit ausreichender Länge benötigt. Am einfachsten erfolgt der Zugriff mit der read()-Methode mit 4 Parametern und mit der write()-Methode mit 4 Parametern. Diese erzeugen auch automatisch die bei der I2C-Übertragung benötigte Start und Stop-Bedingung (wenn repeated=false).

Fragen zu I2C mit MBED

- Welche Pins eignen sich für den I2C Bus?
- Wie kann die Übertragungsgeschwindigkeit von 100 kBit auf 400 kBit erhöht werden?
- Wie kann das Datenbyte 0x55 an die I2C-Adresse 0x39 geschrieben werden?
- Wie kann ein einzelnes Datenbyte von der I2C-Adresse 0x39 gelesen werden?



Programmbeispiel für einen I2C-Adress Scanner (14a_I2C_AdressScan)

```
#include "mbed.h"
Serial pc(SERIAL_TX, SERIAL_RX);
I2C i2c(I2C_SDA, I2C_SCL); // PB_9, PB_8

int main() {
    for (int addr=0; addr<128; addr++) {
        // Achtung: mbed will nach der 7 Bit-I2C-Adresse
        // gleich noch das R/W Bit
        int ack = i2c.write(addr<<1, NULL, 0);
        if (ack == 0) {
            pc.printf("I2C device detected at address = %2Xh\n", addr);
        }
    }
    while (1) {}
}
```

Hinweis:

Die hier verwendete Variante von write() erzeugt automatisch die Start-und Stop-Bedingung

— Variante für einen I2C-Adress Scanner (14b_I2C_AdressScan)

```
#include "mbed.h"
Serial pc(SERIAL_TX, SERIAL_RX);
I2C i2c(I2C_SDA, I2C_SCL); // PB_9, PB_8

int main() {
    for (int addr=0; addr<128; addr++) {
        // Achtung: mbed will nach der 7 Bit-I2C-Adresse
        // gleich noch das R/W Bit
        i2c.start();
        int success = i2c.write(addr<<1);
        i2c.stop();
        if (success == 0) {
            pc.printf("I2C device detected at address = %2Xh\n", addr);
        }
    }
    while (1) {}
}
```

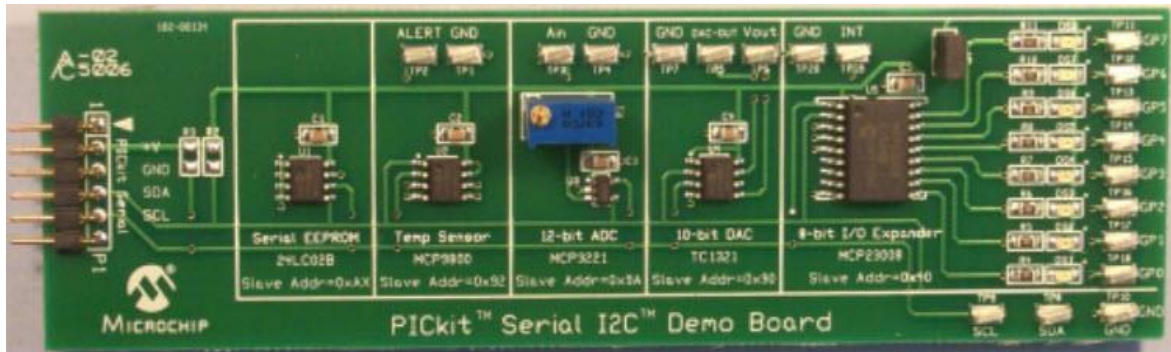
Hinweis:

Bei dieser Variante von write() wird nur ein Byte an den I2C-Slave geschrieben. (Dabei könnte es sich um eine Adresse mit Read/Write-Bit oder auch um ein Datenbyte handeln)

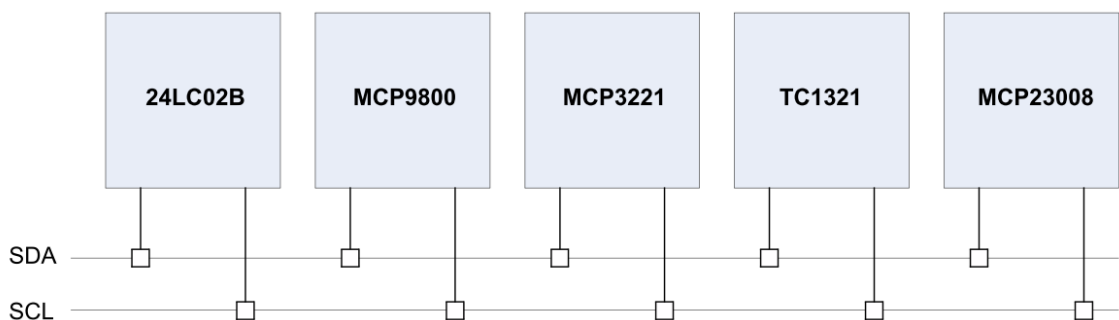
Deshalb muss die Start-und Stop-Bedingung extra erzeugt werden.



Das I2C DEMO Board von Microchip.



Die Adressierung von I2C-Bausteinen erfolgt über eine 7-Bit-Adresse (gefolgt vom Read/Write-Bit). Meist werden 4 Adress-Bits davon durch den Hersteller festgelegt, die weiteren 3 Adress-Bits können dann durch elektronische Beschaltung vom Entwickler festgelegt werden.



- a) Ermitteln Sie die 7-Bit-Adressen, mit denen die I2C-Bausteine aus dem Programm heraus erreichbar sind, durch Analyse von Datenblättern und Schaltplan der I2C-Demo-Platine.

Baustein	Typ	A6	A5	A4	A3	A2	A1	A0	in Hex	<< 1
Serial EEPROM	24LC02									
Temperatur Sensor	MCP9800									
12-Bit A/D Converter	MCP3221									
10-Bit D/A Converter	TC1321									
8-Bit I/O Expander	MCP23008									

- b) Wofür stehen die Abkürzungen?
SDA →
SCL →
- c) Welche Ruhepegel können auf den Leitungen SDA und SCL gemessen werden?
- d) Wie groß sollten die Pull-Up-Widerstände zwischen SDA und SCL nach VCC sein?
- e) Die Datenübertragung auf dem I2C-Bus beginnt immer mit der **Startbedingung**. Wie sieht diese aus?
- f) Die Datenübertragung auf dem I2C-Bus endet mit der **Stopbedingung**. Wie sieht diese aus?

 F. A. HASELWANDER GEWERBLICH-TECHNISCHE SCHULEN OFFENBURG	e-MBED-ded Programmierung	Klasse:
	Name:	Datum:

Zugriff auf den Analog nach Digital-Wandler MCP3221

- Welche Auflösung besitzt der ADC?
- Wie viele Messwerte können maximal in einer Sekunde erfasst werden?
- Wie lange dauert die Datenübertragung für einen Messwert wenn der Zugriff im Fast Mode mit 400Khz erfolgt?
- Am AIN-Eingang des ADC wird eine Spannung von 1,0V angelegt. Welchen digitalen Wert liefert der ADC bei einer Versorgungsspannung von 3,3V
- Schreiben Sie ein Programm (13d_I2C_ADC_MCP3221), welches alle Sekunden den AD-Wert erfasst. Übertragen Sie den gemessenen Wert in Hex und umgerechnet als Spannung an den PC.

Zugriff auf den Digital nach Analog-Wandler TC1321

- Wofür dient der Baustein MCP1524 auf der I2C-Demo-Platine?
- Wodurch unterscheiden sich die Pins DAC-Out und Vout?
- Wie kann der DAC in den Standby Modus versetzt werden?
- Welchen Werte müssen Sie in die Datenregister schreiben um an Vout eine Spannung von 1V zu erzielen?
- Schreiben Sie ein Programm (13e_I2C_DAC_TC1321_Saegezahn) das eine Sägezahnspannung (von 0V - 1V) mit einer Frequenz von ca. 100Hz erzeugt. Für diese Anwendung ist es ausreichend, wenn Sie dabei nur die Datenbits D9 bis D2 verändern.
- Schreiben Sie ein Programm (13f_I2C_DAC_TC1321_Sinus), das einen 50Hz Sinus (mit DC-Anteil von 1,25V und Amplitude von 1,25V) auf dem Vout des DAC ausgibt.
Legen Sie hierfür eine Sinus-Tabelle mit 24 Werten (10Bit) an (siehe Excel-Sheet).
In der ISR eines Tickers, soll immer wieder in die Variable newValue der Wert true geschrieben werden. Werten Sie diesen in der Endlosschleife des Hauptprogramms aus.
Geben Sie auf dem DAC einen neuen Wert aus der Tabelle aus, wenn in der Variable true steht. Löschen Sie diese Merker-Variable nachdem der DA-Wert ausgegeben wurde.

Zugriff auf den IO-Expander MCP23008

- a) Schreiben Sie ein Programm (13g_I2C_IO-Expander_MCP23008_Lauflicht) das auf dem IO-Expander Baustein ein Lauflicht ausgibt.
 Schreiben Sie zunächst ein Unterprogramm `initI2C()`, das die IOs des MCP23008 als Ausgänge konfiguriert.
 In der ISR eines Tickers erhält die globale Variable `newValue` alle 0,5s den Wert `true`.
 Werten Sie diese Variable in der Endlosschleife des Hauptprogramms aus.
 Ist diese `true` so soll ein neuer Lauflicht-Wert auf die Ausgänge des IO-Expanders ausgegeben werden.
 Schreiben Sie danach `false` in die Variable.
- b) Acht an die IOs des IO-Expanders MCP23008 angeschlossene Schalter/Taster sollen per Programm (13h_I2C_IO-Expander_MCP23008_Read) zyklisch alle Sekunde gelesen werden.
 Schreiben Sie zunächst ein Unterprogramm `initI2C()`. Darin sollen die IOs des Bausteins als Eingänge konfiguriert werden. Alle low-aktiven Eingangssignale sollen bereits im Baustein invertiert werden.
 Zudem sind die internen Pull-Up Widerstände zu aktivieren.
 Lesen Sie nun in der Endlosschleife die Input-Werte vom IO-Expander ein. Geben Sie diese auf PC7 bis PC0 bzw. auf der SPI-basierten 8-fach-Siebensegmentanzeige aus.
- c) Machen Sie sich mit den Interruptmöglichkeiten des MCP23008 vertraut. Es soll in einem Programm (13i_I2C_IO-Expander_MCP23008_Interrupt) die Interrupt-Möglichkeiten des Bausteins genutzt werden.
 Initialisieren Sie den Baustein im Unterprogramm `initI2C()` wie folgt.
- deaktivieren Sie den Sequential-Mode, ODR soll aktiv sein, INTPOL high aktiv
 - alle PortPins sind Eingangssignale
 - aktivieren Sie die PullUp-Widerstände für jeden IO
 - der erwartete Default-Value an den Eingangssignalen sollte 0b00001111 sein
 - aktivieren Sie für jeden IO-Pin Interrupt on change
 - bei einer Abweichung gegenüber dem Default-Value soll ein Interrupt ausgelöst werden
- Jetzt sollten Sie eigentlich den INT-Pin des MCP23008 mit dem Port-Pin PC_08 des MC verbinden.
 Sobald ein externer Interrupt an PC_08 erkannt wird, soll die Variable `newValue` gesetzt werden.
 Werten Sie diese in der Endlosschleife des Hauptprogramms aus. Übertragen Sie den zum Unterbrechungszeitpunkt im Register `INTCAP` des MCP23008 gespeicherten Wert über die serielle Schnittstelle an den PC.

Temperaturmessung und Thermostat mit MCP9801

- a) Der I2C-Baustein MCP9801 kann zur Temperaturmessung und als Thermostat genutzt werden. Diesen wollen per Programm (13j_I2C_Temperatur_Sensor_MCP9801) ansteuern.
 Initialisieren Sie den Baustein im Unterprogramm `init_MCP9801()` wie folgt.
- One-Shot disabled, 12Bit Auflösung, default fault queue, alert active-high, comparator-mode, shutdown disabled
 - Die obere Grenztemperatur für den Thermostat beträgt 28°C
 - Die untere Grenztemperatur für den Thermostat soll 26°C betragen.
- Messen Sie die Temperatur in der Endlosschleife des Hauptprogramms alle Sekunde. Übertragen Sie die gemessene Temperatur an den PC.
- b) Wodurch unterscheidet sich der Comparator Mode vom Interrupt-Mode
- c) Was hat es mit der fault queue auf sich?