I2C Bus Functions	I2C Funktionen
The I2C Functions are intended for easy interfacing between C programs and various peripherals using the Philips I2C bus. These functions treat the microcontroller as a bus master and the peripherals as slaves. The prototypes for these functions are placed in the file i2c.h, located in the ..\INC subdirectory. This file must be #include -ed before using the functions. Prior to #include -ing the i2c.h file, you must declare which microcontroller port and port bits are used for communication through the I2C bus.	Die I2C Funktionen werden für ein leichtes Interfacing zwischen C Programmen und verschiedenen Peripheriegeräten mit Hilfe des Philips I2C Busses definiert. Diese Funktionen behandeln den Mikrocontroller als einen Busmaster und die Peripheriegeräte als Slaves. Die Prototypen für diese Funktionen in sind im File i2c.h . .\Unterverzeichnis INC platziert. Diese Akte muss vor dem Verwenden #include -ed der Funktionen sein. Vor dem #include -ing des i2c.h Files, müssen Sie definieren, welcher Mikrocontrollerport und welche Portbits die für Kommunikation durch den I2C Bus verwendet werden.
Example:	Beispiel:
* the I2C bus is connected to PORTB */ /* the SDA signal is bit 3 */ /* the SCL signal is bit 4 */ #asm .equ __i2c_port=0x18 .equ __sda_bit=3 .equ __scl_bit=4 #endasm /* now you can include the I2C Functions */ #include <i2c.h> The I2C Functions are: void i2c_init(void) this function initializes the I2C bus. This is the first function that must be called prior to using the other I2C Functions. unsigned char i2c_start(void) issues a START condition. Returns 1 if bus is free or 0 if the I2C bus is busy. void i2c_stop(void) issues a STOP condition. unsigned char i2c_read(unsigned char ack) reads a byte from the bus. The ack parameter specifies if an acknowledgement is to be issued after the byte was read. Set ack to 0 for no acknowledgement or 1 for acknowledgement. unsigned char i2c_write(unsigned char data) writes the byte data to the bus.	/* Der I2C Bus ist mit PORTB verbunden*/ /* das SDA Signal ist Bit 3 */ /* das SCL Signal ist Bit 4 */ #asm .equ __i2c_port=0x18 .equ __sda_bit=3 .equ __scl_bit=4 #endasm /* nun können Sie die I2C Funktionen einbinden */ #include <i2c.h> /*die I2C Funktionen sind:*/ void i2c_init(void) /*diese Funktion initialisiert den I2C Bus. Dies ist die erste Funktion, die vor dem Verwenden der anderen I2C Funktionen angerufen werden muss.*/ unsigned char i2c_start(void) /*gibt eine Start Kondition aus. gibt 1 zurück wenn Bus frei ist, oder 0 wenn der I2C Bus „busy“ ist .*/ void i2c_stop(void) /*gibt eine Stop Kondition aus.*/ unsigned char i2c_read(unsigned char ack) /*liest ein Byte aus dem Bus. Der „ack“ Parameter gibt an, ob eine Bestätigung ausgegeben werden soll, nachdem das Byte gelesen wurde. Setzen Sie ack auf 0 für keine Bestätigung oder 1 für Bestätigung. */ unsigned char i2c_write(unsigned char data) /*schreibt die Byte-Daten auf den Bus. Gibt 1 zurück, wenn der Sklave bestätigt, oder

Returns 1 if the slave acknowledges or 0 if not.	0 wenn nicht.*/
Example how to access an Atmel 24C02 256 byte I2C EEPROM:	Beispiel wie man kann auf ein Atmel 24 C02 256 Byte I2C EEPROM zugreift:

/* the I2C bus is connected to PORTB */
/* the SDA signal is bit 3 */
/* the SCL signal is bit 4 */
#asm
.equ __i2c_port=0x18
.equ __sda_bit=3
.equ __scl_bit=4
#endasm
/* now you can include the I2C Functions */
 	
#include <i2c.h>
/* function declaration for delay_ms */
#include <delay.h>
#define EEPROM_BUS_ADDRESS 0xa0
/* read a byte from the EEPROM */
unsigned char eeprom_read(unsigned char address)
{
unsigned char data;
i2c_start();
i2c_write(EEPROM_BUS_ADDRESS);
i2c_write(address);
i2c_start();
i2c_write(EEPROM_BUS_ADDRESS | 1);
data=i2c_read(0);
i2c_stop();
return data;
}
/* write a byte to the EEPROM */
void eeprom_write(unsigned char address, unsigned char data)
{
i2c_start();
i2c_write(EEPROM_BUS_ADDRESS);
i2c_write(address);
i2c_write(data);
i2c_stop();
/* 10ms delay to complete the write operation */
delay_ms(10);
}
void main(void)
{
unsigned char i;
/* initialize the I2C bus */
i2c_init();
	/* der I2C Bus ist mit PORTB verbunden */
	/* das SDA Signal ist Bit 3 */
	/* das SCL Signal ist Bit 4 */
	#asm
	.equ __i2c_port=0x18
	.equ __sda_bit=3
	.equ __scl_bit=4
	#endasm
	/* jetzt Sie können die I2C Funktionen einschließen */
	#include <i2c.h>
	/* Funktionserklärung für delay_ms */
	#include <delay.h>
	#define EEPROM_BUS_ADDRESS 0xa0
	/* lesen eines Bytes aus dem EEPROM */
	unsigned char eeprom_read(unsigned char address)
	{
	unsigned char data;
	i2c_start();
	i2c_write(EEPROM_BUS_ADDRESS);
	i2c_write(address);
	i2c_start();
	i2c_write(EEPROM_BUS_ADDRESS | 1);
	data=i2c_read(0);
	i2c_stop();
	return data;
	}
	/* schreiben eines Bytes auf das EEPROM */
	void eeprom_write(unsigned char address, unsigned char data)
	{
	i2c_start();
	i2c_write(EEPROM_BUS_ADDRESS);
	i2c_write(address);
	i2c_write(data);
	i2c_stop();
	/* 10 ms Verzögerung, um die Schreiboperation zu beenden */
	delay_ms(10);
	}
	void main(void)
	{
	unsigned char i;
	/* initialisieren Sie den I2C Bus */
	i2c_init();

/* write the byte 55h at address AAh */
 	
eeprom_write(0xaa,0x55);
/* read the byte from address AAh */
i=eeprom_read(0xaa);
while (1); /* loop forever */
}
	/* schreiben des Bytes 55 h nach Adresse AAh */
	eeprom_write(0xaa,0x55);
	/* lesen des Bytes aus Adresse AAh */
	i=eeprom_read(0xaa);
	while (1); /* Endlosschleife */
	}