Praktikums-Termine (1 – 4)

Versuchsinhalte	Teilversuche
1. *Einführung* Kennenlernen der Hardware und Entwicklungs-Umgebung	• Einführung • Blinklicht in Assembler • Sekundenzähler mit der 7-Segment-Anzeige in C
2. *Timer-/Interrupt-Behandlung und Parallele Ein-/Ausgänge* Kennenlernen der Timer- und Interrupt-Nutzung kombiniert mit dem Scannen eines 3x3 Tastenfeldes	• Timer- und Interrupt-Nutzung zur LED-Anzeige als Sekundenzähler • Tastenfeld-Abfrage mit Ziffern-Darstellung auf der 7-Segment-Anzeige
3. *ADU-Nutzung und USART-Nutzung zur Temperaturwert-Erfassung* Ansteuerung des Analog-Digital-Umsetzers zur Bestimmung der Potentiometerstellung, zur Temperaturwert-Messung und zur Übertragung der Messwerte an einen PC inklusive Wertdarstellung	• Poti-Spannungsabfrage und LED-Anzeige eines proportionalen Wertes • Spannung an NTC erfassen, in Temperatur umrechnen und per USART als Textzeichenfolge übertragen
4. *Steuerung eines PPG-Sensormoduls über die I2C-Schnittstelle* Ansprechen, konfigurieren und abfragen eines integrierten PPG-Sensors (MAX30102) über die I2C-Schnittstelle, Pulsberechnung und Übertragung der Messdaten an einen PC inklusive Wertdarstellung	• PPG-Sensor per I2C konfigurieren, auslesen und per USART als Textzeichen-Graph übertragen • Herzfrequenz aus PPG-Wertefolge berechnen und per USART als Textzeichenfolge übertragen

Inhalt

1 Grundlagen der Photoplethysmographie

Die Photoplethysmographie ist ein optisches Verfahren zur Messung von zeitlich veränderlichen Gewebezuständen eines Körperteils oder Organs. Hierbei wird ein räumlich begrenztes Hautareal einer Strahlung definierter Wellenlängen (600 – 1000 nm) ausgesetzt. Hämoglobin absorbiert die emittierte Strahlung wesentlich stärker als das umgebende Gewebe. Dessen Konzentration und Volumen im Messareal bestimmt im Wesentlichen den Momentanwert des absorbierten Anteils der Strahlung und lässt sich über den reflektierten bzw. transmittierten Lichtanteil nachweisen. Die am weitesten verbreitete Anwendungsform des PPG-Verfahrens ist die Pulsoxymetrie, bei der nach diesem Verfahren an einer Fingerspitze oder dem Ohrläppchen der Puls und die Sauerstoffsättigung gemessen werden kann. Mit jedem Druckpuls, der vom Herzen, durch das arterielle Gefäßsystem des Körpers, bis in die feinsten Kapillaren der Haut wandert, dehnen sich die Gefäße lokal kurzzeitig aus. Diese Volumenänderung verursacht auch eine Änderung in der Absorption des Lichtes im Gewebe und ist damit mittels der Photoplethysmographie messbar.

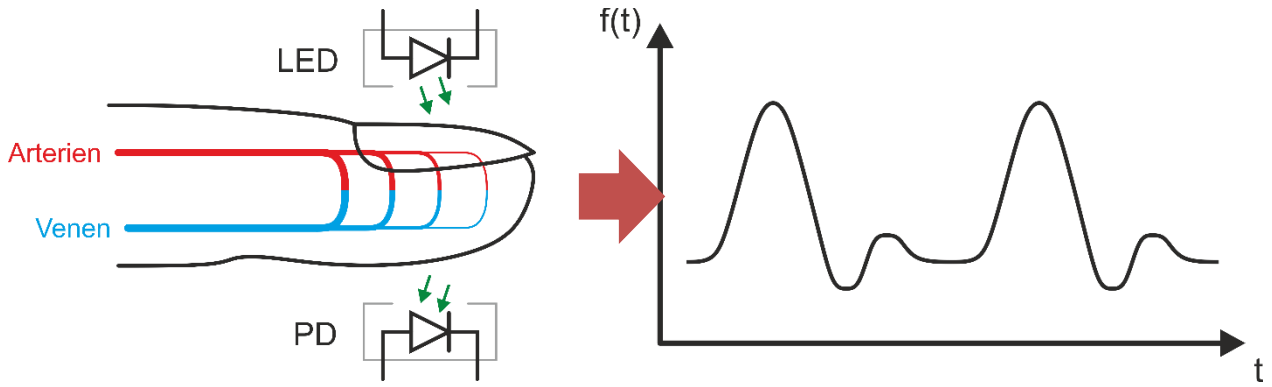

Abbildung 1: Pulsmessung mittels der Photoplethysmographie (PPG)

Soll nur der Puls erfasst werden, reicht für die Beleuchtung eine einzelne LED aus, soll dagegen auch die relative Sauerstoffsättigung gemessen werden, kommen zwei LEDs mit unterschiedlichen Wellenlängen (z. B. rot und infrarot) im ständigen Wechsel zum Einsatz. Befindet sich das untersuchte Körpergewebe zwischen Lichtquelle und Photodiode erfolgt die Messung anhand des transmittierten Lichtes. Befinden sich die optoelektronischen Bauteile dagegen nebeneinander angeordnet und sind parallel auf die Haut, auf der der Sensor aufliegt, ausgerichtet, wird für die Messung das reflektierte Licht erfasst.

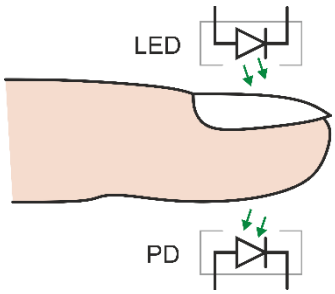

Abbildung 2: PPG in Transmission

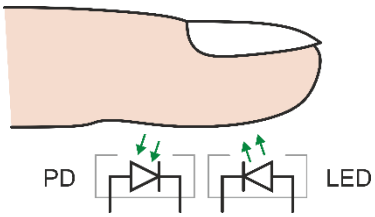

Abbildung 3: PPG in Reflexion

2 Aufbau und Funktion des integrierten PPG-Sensormoduls MAX30102

Abbildung 4: PPG-Sensormodul MAX30102

Im Praktikum wird ein integriertes PPG-Sensormodul der Firma MAXIM mit den Hauptabmessungen 5,6 x 3,3 x 1,55 mm^3 verwendet (s. Ansichten in Abbildung 4), das alle für die Ableitung von PPG-Signalen erforderlichen optischen und für die Signalverarbeitung erforderlichen Schaltungs-Komponenten enthält. Das Modul arbeitet nach der Reflexionsmethode und besteht aus zwei separaten verschieden-farbigen LEDs, einem Photodetektor und einer analogen Verstärker- und Filterschaltung. Darüber hinaus ist ein Analog-Digital-Umsetzer mit einer digitalen Inter-IC-Bus-Schnittstelle (I2C) enthalten, der die analogen verstärkten Photodetektorspannungen digitalisiert und über das Bus-System allen am Bus angeschlossenen Teilnehmern (hier unserem ATmega328PB) zur Verfügung stellt. Über die Busschnittstelle ist das Modul auch konfigurierbar, wobei unter anderem der Betriebsmodus (Pulsmessung, Sauerstoffsättigungsmessung, etc.), die Helligkeit der LEDs und die Messwiederholrate konfigurierbar sind. Eine der beiden LEDs emittiert rotes Licht mit einer mittleren Wellenlänge von 660 nm, die andere infrarotes Licht mit einer mittleren Wellenlänge von 880 nm. Die LEDs werden je nach Betriebsmodus und Messwiderholrate durch die im Modul integrierte Ablaufsteuerung automatisch an- und abgeschaltet. Der integrierte Photodetektor ist von seiner Empfindlichkeitskennlinie auf die beiden LED-Wellenlängen abgestimmt und damit für beide Mittenwellenlängen annähernd gleich empfindlich. Der vom Detektor gelieferte Strom ist über einen großen Bereich nahezu proportional zur Beleuchtungsstärke. Die Verschaltung und der innere Aufbau des PPG-Moduls sind in Abbildung 5 gezeigt. Erkennbar sind weitere Funktionsmodule, die jedoch in diesem Praktikum nicht relevant sind.

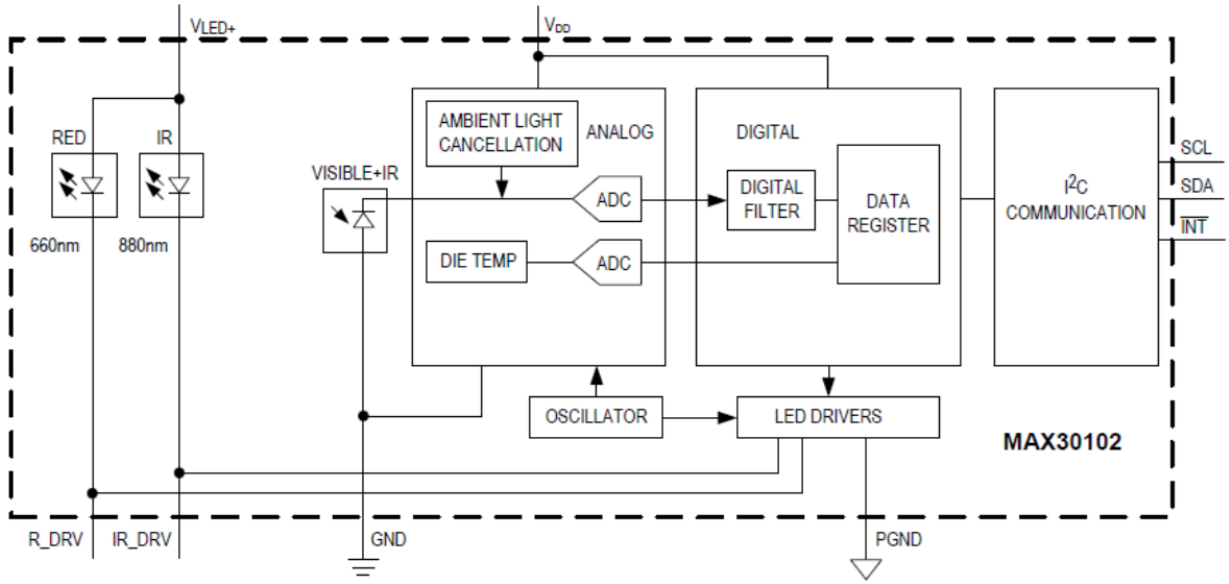

Abbildung 5: Blockstruktur des verwendeten Pulssensors

Der Eingebaute ADU hat eine Auflösung von 18 Bit und erlaubt eine Messrate der PPG-Signalwerte von 50 sps (*samples per second*) bis zu 3200 sps. Für Pulsmessungen reichen in der Regel 50 sps aus.
Die Konfiguration des PPG-Moduls und das Auslesen der PPG-Daten erfolgt über interne Register, die über den I2C-Bus gelesen und beschrieben werden können. Jede I2C-Buskomponente muss eine am Bus eindeutige 8-Bit-Adresse besitzen (strenggenommen sind nur die höchstwertigen 7 Bit des 8-Bit-Adresswortes die Adresse, da das 0'te Bit die Art des Zugriffs, also lesen oder schreiben kennzeichnet). Im Fall des MAX30102 ist dies die **Adresse 0xAE**. Über diese Adresse kann in Verbindung mit einer nachfolgend gesendeten Registernummer der Inhalt des betreffenden Registers gelesen, oder durch senden eines weiteren Datenwortes mit diesem beschrieben werden. Der Baustein verwendet 20 derzeit nutzbare Registeradressen im Adressbereich 0x00 bis 0xFF. Einen vollständigen Überblick zu den Registerdressen und -bedeutungen gibt die Tabelle der Register im Anhang.
Nachfolgend sind die im Praktikum genutzten Register und deren genutzte Inhalte (rote Rahmen) in der Reihenfolge ihrer Nutzung beschrieben:

Mode Configuration (0x09)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
Mode Configuration	SHDN	RESET				MODE[2:0]			0x09	0x00	R/W

Über dieses Konfigurationsregister wird der Betriebsmodus des PPG-Moduls eingestellt. Grundsätzlich sind die Messabläufe automatisiert und können daher vom Nutzer nicht selber in Einzelschritten durchgeführt werden. Je nach Konfigurationsvorgaben werden die benötigten LEDs je Messperiode kurz aktiviert, die Detektorspannung wird gemessen, gefiltert, digitalisiert, digital prozessiert und zur Abholung bereitgestellt. Dieser Prozess wiederholt sich mit weiteren LEDs und wählbarer Wiederholrate. Über die drei Bits Mode[2:0] kann dieser grundlegende Ablauf durch Wahl von einem von drei möglichen Messprofilen vorgegeben werden,
Mode[2:0] = 2 aktiviert den **Herzraten-Modus** (verwendet nur die rote LED),
Mode[2:0] = 3 aktiviert den **SPO2-Modus** (verwendet rote und infrarote LED im Wechsel),
Mode[2:0] = 7 aktiviert den **Multi-LED-Modus** (wie SPO2-Modus aber ohne Filterung).

FIFO Configuration (0x08)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
FIFO Configuration	SMP_AVE[2:0]			FIFO_ROLLOVER_EN	FIFO_A_FULL[3:0]				0x08	0x00	R/W

Je nach Messprofil wird je Messzyklus eine der Messauflösung entsprechende Anzahl von Datenworten (Bytes) als Messergebnis generiert und in einem internen Pufferspeicher (FIFO) abgelegt. Im Herzraten-Modus sind das 3 Byte, im SPO2- und Multi-LED-Modus sind das 6 Byte anfallender Daten je Zyklus. Beim Auslesen des Pufferspeichers wird eine entsprechende Anzahl von Speicherstellen im Pufferspeicher wieder freigegeben. Wenn der Pufferspeicher nicht ausgelesen wird, füllt er sich mit jedem weiteren Messzyklus, bis er vollständig vollgeschrieben ist.
Über das Bit FIFO_ROLLOVER_EN im FIFO-Konfigurationsregister wird festgelegt, was im Fall eines überlaufenden Pufferspeichers passieren soll. Ist dieses Bit "0", dann wird das

aktuelle Messergebnis nicht abgespeichert und geht verloren. Ist das Bit "1", dann werden die ältesten Einträge im Pufferspeicher mit den neuen Messdaten überschrieben ("***Rollover***").
Zur Erhöhung der Messgenauigkeit durch Reduzierung der Rausch- und Quantisierungsfehler kann das sogenannte "***Oversampling***" angewendet werden. Dabei werden die Messergebnisse einer Folge von Messzyklen zu einem Mittelwert zusammengefasst. Nur dieser Mittelwert über n Messwerte wird in den Pufferspeicher geschrieben und dadurch verfügbar gemacht. Über die drei Bits SMP_AVE[2:0] wir die Zahl n festgelegt. Der Eintragswert entspricht dem Exponenten des Zweierpotenzwertes von n (Beispiel: SMP_AVE[2:0] = 4 bedeutet $n = 2^4$, also ein Mittelwert über 16 Messwerte). Es kann maximal ein Mittelwert über 32 Messwerte gebildet werden (SMP_AVE[2:0] = 5) oder die Mittelwertbildung kann deaktiviert werden (SMP_AVE[2:0] = 0).

SpO_2 Configuration (0x0A)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
SpO_2 Configuration		SPO2_ADC_RGE[1:0]		SPO2_SR[2:0]			LED_PW[1:0]		0x0A	0x00	R/W

Über dieses Konfigurationsregister lassen sich die Messdauer (Pulsweite der LED-Pulse für beide LED-Typen gemeinsam) und die Messwiederholrate in definierten Schritten einstellen.
Mit den zwei Bits LED_PW[1:0] kann die **Messdauer** aufsteigend in den vier Stufen 69 µs, 118 µs, 215 µs und 411 µs gewählt werden, wobei die Messdauerwahl (und damit die Integrationszeit des ADU) mit der Messauflösung korreliert ist und zu den Auflösungen von 15 Bit (bei 69 µs) bis 18 Bit (bei 411 µs) passen.
Die drei Bits SPO2_SR[2:0] bestimmen die **Messwiederholrate**, die darüber aufsteigend in den 8 sps-Stufen 50, 100, 200, 400, 800, 1000, 1600, 3200 einstellbar ist.
ACHTUNG! Messwiederholrate und Messdauer bedingen einander und sind nicht beliebig frei wählbar (Beispiel: 3200 sps im SPO2-Modus mit zwei verschiedenen LED-Typen im Wechsel bedeuten einen Abstand zwischen zwei Messungen derselben LED von 312,5 µs was für die Messdauer einen maximalen Wert je LED-Typ von 156,25 µs bedeutet und damit nur die maximale Messdauerstufe von 118 µs zulässt).

LED Pulse Amplitude (0x0C–0x0D)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
LED Pulse Amplitude	LED1_PA[7:0]								0x0C	0x00	R/W
	LED2_PA[7:0]								0x0D	0x00	R/W

Mit den acht Bits LED1_PA[7:0] dieser beiden Register kann für die rote LED und mit LED2_PA[7:0] für die infrarote LED der **Treiberstrom** und damit die Lichtintensität der jeweiligen LED in Stufen von 0,2 mA im Wertebereich von 0 mA bis 51 mA eingestellt werden. Mit zunehmender Lichtintensität der LEDs steigt die Empfindlichkeit der PPG-Messung aber gleichzeitig auch die Stromaufnahme des PPG-Moduls und damit der Energieverbrauch. In der vorliegenden Anwendung im Versuch 4 sollte der LED-Strom unter 10 mA bleiben um den zur Versorgung des PPG-Moduls genutzten Port-Pin PE2 nicht zu überlasten.

Interrupt Enable (0x02-0x03)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
Interrupt Enable 1	A_FULL_EN	PPG_RDY_EN	ALC_OVF_EN						0x02	0X00	R/W
Interrupt Enable 2							DIE_TEMP_RDY_EN		0x03	0x00	R/W

Das PPG-Modul verfügt über eine eigene stabilisierte Taktversorgung als Basis für alle davon abgeleiteten Zeiten der Messabläufe. Um diese mit den Signalverarbeitungsabläufen eines angeschlossenen Mikrocontrollers (wie unseres ATmega328PB) zu synchronisieren, kann das PPG-Modul bei Vorhandensein neuer Messdaten im Pufferspeicher einen **Interrupt** auslösen. Hierzu existiert am PPG-Modul ein Interrupt-Ausgangssignalanschluss, der mit einem Interrupteingang eines Mikrocontrollers verbunden werden kann. Wegen der beschränkten Anschlussmöglichkeiten des im Praktikum verwendeten Controller-Boards ist dies hier leider nicht erfolgt. Trotzdem kann der Interrupt-Mechanismus des PPG-Moduls nach Aktivierung über das Bit PPG_RDY_EN im Interrupt-Enable-Register genutzt werden, um im Zyklus der Messwiederholrate ein abfragbares Triggerereignis zu generieren (s. Interrupt-Status-Register)

Interrupt Status (0x00–0x01)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
Interrupt Status 1	A_FULL	PPG_RDY	ALC_OVF					PWR_RDY	0x00	0X00	R
Interrupt Status 2							DIE_TEMP_RDY		0x01	0x00	R

Dieses Register enthält das auslesbare Bit PPG_RDY, dass nach Aktivierung des zuvor beschriebenen Interrupt-Mechanismus des PPG-Moduls immer dann automatisch auf den Wert "1" gesetzt wird, wenn ein neues Messergebnis im Pufferspeicher vorliegt und wieder auf den Wert "0" zurückgesetzt wird, wenn die Daten des Pufferspeichers durch den angeschlossenen Mikrocontroller ausgelesen wurden. Der Mikrocontroller muss dazu in häufiger Folge das PPG_RDY-Bit abfragen ("***Polling***") um diesen Statuswechsel zu erkennen.

FIFO (0x04–0x07)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
FIFO Write Pointer				FIFO_WR_PTR[4:0]					0x04	0x00	R/W
Over Flow Counter				OVF_COUNTER[4:0]					0x05	0x00	R/W
FIFO Read Pointer				FIFO_RD_PTR[4:0]					0x06	0x00	R/W
FIFO Data Register	FIFO_DATA[7:0]								0x07	0x00	R/W

Der **Pufferspeicher** (FIFO = "*first-in-first-out*") des PPG-Moduls kann die Messdaten aus 32 Messzyklen im SPO2-Modus aufnehmen. Dies entspricht bei 18 Bit Messauflösung

(erfordert 3 Byte je Messwert) und zwei Messungen je Zyklus (rote und infrarote LED) einem Tupel von 6 Byte je Messzyklus, bei 32 Tupeln also einer Datenmenge von 192 Bytes. Die Anordnung der 18 Bit je Messwert in den dadurch jeweils belegten 3 Byte des Pufferspeichers ist dabei wie folgt:

BYTE 1							FIFO_ DATA[17]	FIFO_ DATA[16]
BYTE 2	FIFO_ DATA[15]	FIFO_ DATA[14]	FIFO_ DATA[13]	FIFO_ DATA[12]	FIFO_ DATA[11]	FIFO_ DATA[10]	FIFO_ DATA[9]	FIFO_ DATA[8]
BYTE 3	FIFO_ DATA[7]	FIFO_ DATA[6]	FIFO_ DATA[5]	FIFO_ DATA[4]	FIFO_ DATA[3]	FIFO_ DATA[2]	FIFO_ DATA[1]	FIFO_ DATA[0]

Eingehende neue Messdaten-Tupel des PPG-Moduls werden über die Register 0x04 bis 0x07 zugänglich gemacht. Register 0x07 dient dabei als Schreib-/Lese-Fenster über das auf genau eines der 192 Bytes des Puffers zugegriffen werden kann. Welches der 192 Bytes das jeweils ist, bestimmt bei Leseoperationen der Lesezeiger FIFO_RD_PTR[4:0] und bei Schreiboperationen der Schreibzeiger FIFO_WR_PTR[4:0], die jeweils auf die Startadresse eines neuen 6-Byte-Tupels zeigen. Nach jeder Lese- oder Schreiboperation wird der entsprechende Zeiger automatisch inkrementiert, so dass z. B. bei aufeinanderfolgenden Leseoperationen auf das FIFO-Daten-Register 0x07 jedes Mal ein anderes Byte aus dem Pufferspeicher gelesen wird. Gleiches gilt übertragend für den schreibenden Zugriff. Wenn beim automatischen Inkrementieren der Zeiger die letzte Registeradresse (Adresse 31) erreicht ist, beginnt mit der nächsten Inkrementierung die Zählung wieder von vorne (Adresse 0). Die Zahl der 6-Byte-Tupel im Pufferspeicher, die noch nicht ausgelesen wurden, ergibt sich als Differenz zwischen dem Wert des Schreibzeigers und dem Wert des Lesezeigers.

Um die zuvor beschriebenen Register des MAX30102 PPG-Moduls zu beschreiben und auszulesen existiert eine Bibliothek von C-Funktionen, die dies auf einfache Art und Weise ermöglichen. Im Folgenden werden die Grundlagen der Buskommunikation über den I2C-Bus und die Nutzung dieser C-Bibliotheksfunktionen beschrieben.

3 Grundlagen und Nutzung der I2C-Schnittstelle

Der I2C-Bus (Inter-IC-Bus) ist ein vergleichsweise einfach aufgebautes Bus-System, das ursprünglich von der Firma Philips (heute NXP) entwickelt wurde, um integrierte intelligente Schaltungskomponenten auf einer Schaltungsplatine mit geringem Verdrahtungsaufwand über kurze Distanzen zu vernetzen. Viele Bauteilehersteller verwenden dieses Bus-System, das ursprünglich für eine Übertragungsrate von bis zu 100 kBit/s konzipiert war (Standard-Modus) mit heute bis zu 5 MBit/s (in den Varianten Fast-Mode, High-Speed-Mode oder Ultra-Fast-Mode). Das MAX30102 PPG-Modul arbeitet mit einer Übertragungsrate von 400 kBit/s (Fast-Mode). Wesentliche Merkmale des I2C-Busses sind:

- Der Bus ist von seiner Auslegung ein Multi-Master-Bussystem, mit Kollisionserkennung (CSMA/CD, s. SMR-Vorlesung) und Arbitrierung, wird in den meisten Fällen (auch hier im Praktikumsversuch 4) aber in einer Master-Slave-Konstellation genutzt
- Die Übertragung der Daten findet als serieller Datenstrom bitweise statt

- Es werden zwei Kommunikationsleitungen verwendet, eine Datenleitung SDA (*serial data line*) und eine Taktleitung SCL (*serial clock line*) für eine synchrone Übertragung
- Es wird positive Logik verwendet (Signalpegel "high" = "1", Signalpegel "low" = "0")
- Nutzdaten werden als Folge von 8-Bit-orientierten Paketen prinzipiell bidirektional übertragen, jedoch im Halbduplex-Verfahren (nicht gleichzeitig)

In Abbildung 6 ist die hier im Praktikum genutzte Bus-Verschaltung des ATmega328PB als Bus-Master und des MAX30102 PPG-Sensors als Bus-Slave gezeigt. Beim I2C-Bus werden die beiden Kommunikationsleitungen SDA und SCL jeweils über einen Pull-Up-Widerstand mit der gemeinsamen Versorgungsspannung verbunden (notwendig für die Arbitrierung). Durch die hier gewählte Master-Slave-Zuordnung ist gewährleistet, dass immer nur der Master eine neue Übertragungssequenz initiieren kann, also agiert, und der Slave stets nur antwortet, also reagiert. Da der Mikrocontroller die einzige agierende Komponente am Bus ist, benötigt er keine Busadresse. Als reagierende Komponente am Bus, besitzt das PPG-Modul eine Adresse (beim MAX30102 ist dies die **Adresse 0xAE**, die vom Chip-Hersteller fest zugeordnet wurde) und über die das Modul am Bus vom Controller angesprochen werden kann.

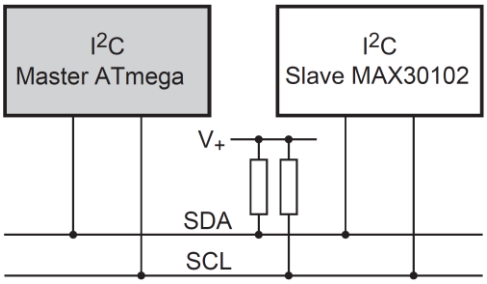

Abbildung 6: Beispielhafte Verschaltung zweier I2C-Bus-Komponenten als Master und Slave

Jede neue Kommunikationssequenz beginnt mit einer *START condition*, die ausgehend vom Leerlaufzustand des Busses (SDA = SCL = "*high*") durch eine fallende Flanke an SDA gekennzeichnet ist. Mit jedem nachfolgenden Taktzyklus an SCL wird ein weiteres Bit des Kommunikationsprotokolls übertragen, wobei die Bedeutung der Bits sich aus deren Position innerhalb des Protokollrahmens ergibt. Abbildung 7 zeigt das Zeitverhalten und die Bedeutung der Signale einer Kommunikationssequenz für die Übertragung von zwei "Nutz-Bytes".

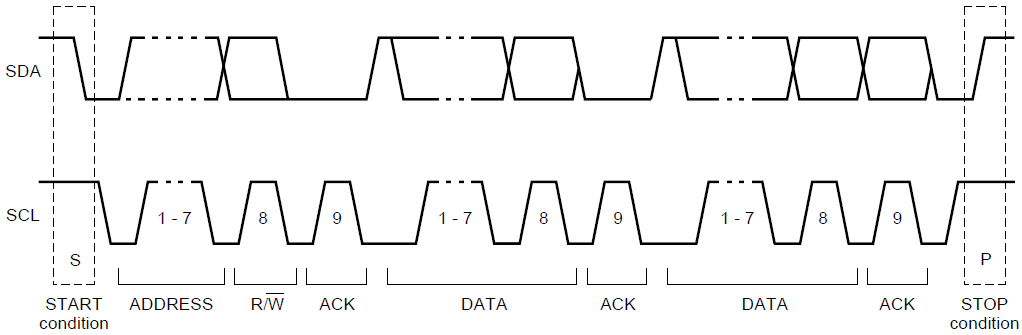

Abbildung 7: Signalzustände und Timing auf den Leitungen SDA und SCK bei der Bus-Kommunikation

Erkennbar ist, dass der Bus-Takt während der ganzen Sequenz mit konstanter Periodendauer aktiv ist und erst am Ende der Sequenz wieder aussetzt (SCL = "*high*"). Wie

auch in Abbildung 8 erkennbar, startet die vom Bus-Master ausgehende Bitfolge zunächst mit den Bits 1 – 7 der Adresse des Slaves, der angesprochen werden soll. Danach folgt ein Bit, das die Datenflussrichtung für die Nutzdatenübertragung angibt ($R/\overline{W}$ = "0" bedeutet lesen, $R/\overline{W}$ = "1" bedeutet schreiben). Der Slave signalisiert mit dem folgenden Bit, ob er für die weitere Übertragung bereit ist (*acknowledge* = "0") oder nicht (not *acknowledge* = "1"). Je nach Datenflussrichtung sendet jetzt der Bus-Master oder -Slave seine Nutzdaten (Bits 0 bis 7) gefolgt von einem erneuten Quittungs-Bit des jeweils anderen Kommunikationspartners (*acknowledge* für korrekt empfangene Daten, *not acknowledge* bei erkannten Übertragungsfehlern). Danach können in gleicher Weise weitere Nutzdaten-Bytes übertragen werden. Jede Kommunikationssequenz endet mit einer *STOP condition*, die durch ein ruhendes SCL-Taktsignal (SCL = "*high*") und durch eine steigende Flanke an SDA gekennzeichnet ist. Hierdurch gelangt der Bus wieder in seinen Leerlaufzustand, aus dem heraus jederzeit eine neue Kommunikationssequenz gestartet werden kann.

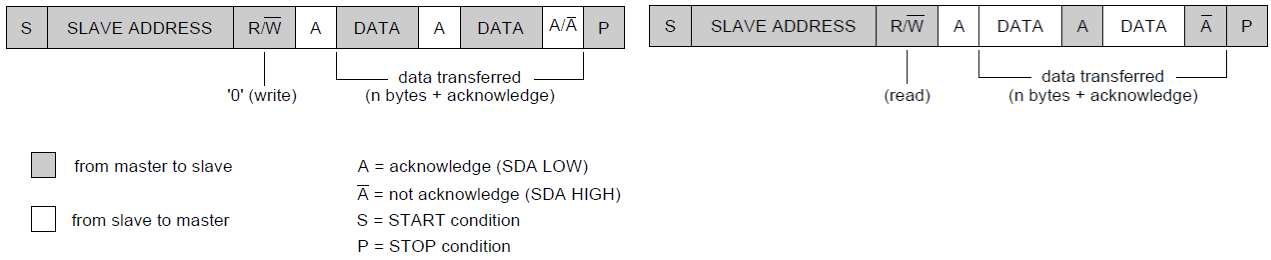

Abbildung 8: Kommunikationsablauf zwischen Sender und Empfänger bei der Bus-Kommunikation für einen sendenden Zugriff (links) und einen lesenden Zugriff (rechts)

Um die programmtechnische Umsetzung der I2C-Buskommunikation zu erleichtern, soll im vorliegenden Versuch die **C-Funktionsbibliothek *i2c.h*** genutzt werden. Sie enthält eine Reihe von Funktionen, mit denen das Initialisieren des Bus-Systems und der lesende und schreiben Zugriff auf die Registerinhalte des MAX30102 stark vereinfacht werden. Nachfolgend werden die hier im Versuch 4 benötigten Funktionen der Bibliothek *i2c.h* hinsichtlich ihrer Anwendung kurz beschrieben.
Die Bibliothek *i2c.h* ist keine C-Standardbibliothek und muss daher zusammen mit *i2c.c* in das Arbeitsverzeichnis des Programmierprojekts kopiert und dem Projekt hinzugefügt werden. Dies erfolgt am einfachsten mit Hilfe des **Solution Explorers**. Per **Rechtsklick** auf den **<Projektnamen>** öffnet sich ein Menü, in dem **Add→Existing Item…** gewählt wird. In dem Fenster, das sich daraufhin öffnet, werden die entsprechenden Dateien ausgewählt. Daraufhin werden *i2c.c* und *i2c.h* im Solution Explorer angezeigt.
Die Einbindung in den Programmcode erfolgt über das bekannte Format: #include "i2c.h".

"**i2c_setup()**"- Funktion

Über diese Funktion aus der Programmbibliothek *i2c.h* wird einmalig bei der Initialisierung für die I2C-Schnittstelle 1 des ATmega328PB die Schnittstellenfunktion der zugehörigen Port-Pins aktiviert und die Kommunikationsleitungen der Schnittstelle werden in den Leerlaufzustand gebracht. Diese Funktion besitzt keine einstellbaren Parameter.

"***i2c_read_byte(<Slave-Adr.>, <Register-Nr.>)***"-Funktion

Diese Funktion liest ein komplettes Daten-Byte, bestehend aus 8 seriell übertragenen Daten-Bits, aus dem bezeichneten Register des Bus-Slaves mit der angegebenen Adresse

aus. Adresse und Registernummer sind im vorzeichenlosen 8 Bit Integer-Format *uint8_t* anzugeben. Der Rückgabewert der Funktion entspricht dem gelesenen Registerinhalt im gleichen Format.

"*i2c_read_n_bytes(<Slave-Adr.>, <Register-Nr.>, <n>), <Zeiger auf Array>*"-Funktion

Diese Funktion liest eine Serie von n Daten-Bytes, bestehend aus je 8 seriell übertragenen Daten-Bits, aus dem bezeichneten Register des Bus-Slaves mit der angegebenen Adresse aus und schreibt diese ab der angegebenen Startadresse in das bezeichnete Datenfeld (Array). Byteanzahl n, Adresse und Registernummer sind im vorzeichenlosen 8 Bit Integer-Format *uint8_t* anzugeben. Das Datenfeld auf das der Schreibzeiger zeigt ist ebenfalls vom Datentyp *uint8_t*. Einen Rückgabewert der Funktion gibt es nicht.
Dieser Befehl ist speziell konzipiert für das Auslesen eines Daten-Tupels aus dem Pufferspeicher eines Bus-Slaves über ein einziges Register.

"*i2c_write_byte(<Slave-Adr.>, <Register-Nr.>, <Wert>)*"-Funktion

Diese Funktion schreibt das übergebene Daten-Byte "Wert", bestehend aus 8 seriell übertragenen Daten-Bits, in das bezeichnete Register des Bus-Slaves mit der angegebenen Adresse. Wert, Adresse und Registernummer sind im vorzeichenlosen 8 Bit Integer-Format *uint8_t* anzugeben. Einen Rückgabewert der Funktion gibt es nicht.

4 Praktischer Teil zum Versuch 4 Teil 1

Im Teilversuch 1 des Versuchs 4 soll das im Abschnitt 2 beschriebene, im Praktikums-Sets integrierte PPG-Modul MAX30102 für die periodische Messung und Digitalisierung von PPG-Signalfolgen genutzt werden. Das Modul ist wie in Abbildung 9 gezeigt über vier Leitungen an den Port E des ATmega328PB angeschlossen. Der Interrupt-Ausgang des Moduls ist nicht mit dem Controller verbunden, so dass die Verfügbarkeit neuer Messwerte nur durch wiederholtes Abfragen des Interrupt-Status ("*Polling*"), wie in Abschnitt 2 beschrieben, ermittelt werden kann. Dafür muss der Interrupt des Moduls zuvor initial aktiviert werden.

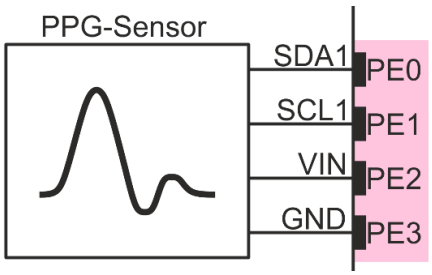

Abbildung 9: Anschluss des PPG-Sensormoduls an den Mikrocontroller

Die Versorgungsspannung bezieht das Modul aus den Port-Pins des Port E. Dazu müssen initial die betreffenden Port-Pins PE3 als Masseanschluss (GND) in den Zustand *OUTPUT LOW* und PE2 als Versorgungsspannung (+5 V) in den Zustand *OUTPUT HIGH* versetzt werden. Da der Mikrocontroller als einziger I2C-Master am Bus fungiert, erfolgt die Taktversorgung für die Kommunikation über die SCL-Leitung an PE1 treibend durch ihn, so dass der betreffende Pin initial in den Leerlaufzustand *OUTPUT HIGH* versetzt werden kann (s. Abbildung 7). Der Kommunikationsanschluss SDA an PE0 sollte sicherheitshalber initial

in den rezeptiven Zustand *INPUT TRISTATE* versetzt werden, damit nach dem Einschalten der Spannungsversorgung durch unbestimmte Signalzustände keine ungewollten Schreib-Aktionen auftreten können.

Die Position des PPG-Moduls unterhalb einer Aussparung im Gehäusedeckel des Praktikums-Sets ist in Abbildung 10 durch einen roten Rahmen markiert. Zur Durchführung der Messung muss ein Finger ohne Andruck wie in Abbildung 11 gezeigt vollflächig auf die Aussparung des Gehäusedeckels gelegt werden, so dass der Finger die nach Aktivierung rot leuchtende LED des Moduls abdeckt.

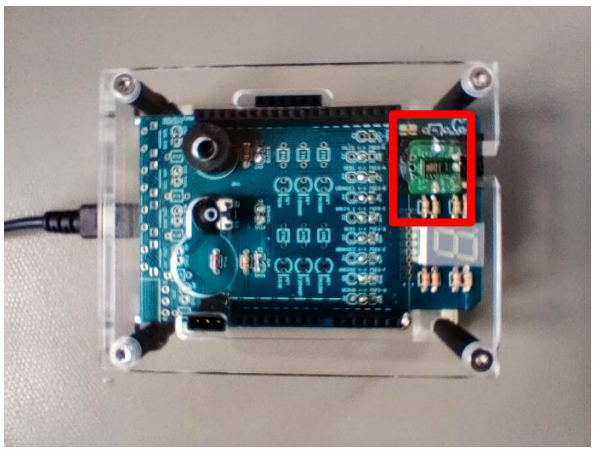

Abbildung 10: Position und Zugang des PPG-Sensors

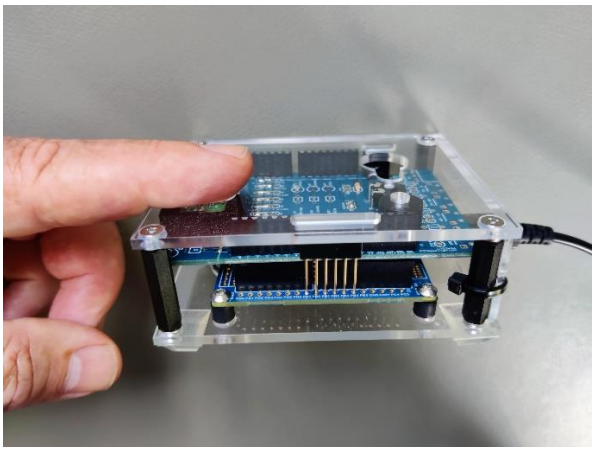

Abbildung 11: Anwendung der PPG-Messung

Die in der gezeigten Messkonstellation gemessenen und digitalisierten 18 Bit PPG-Daten sind durch einen hohen Gleichanteil (Offset) mit überlagertem kleinen pulsabhängigen Wechselanteil gekennzeichnet (siehe auch Abbildung 13). Der Wert des Gleichanteils kann sich jedoch bedingt durch äußere Einflüsse auf die Messung (z. B. Fremdlicht), als auch durch physiologisch bedingte Einflüsse (z. B. Vasokontraktion) mit der Messdauer ändern. Um den reinen pulsatorischen Anteil aus den Messwerten zu extrahieren wird ein nachgeführter Mittelwert über mehrere Signalperioden benötigt, der als Referenzwert vom Momentanwert der Messung abgezogen wird und so den reinen Wechselanteil ergibt.

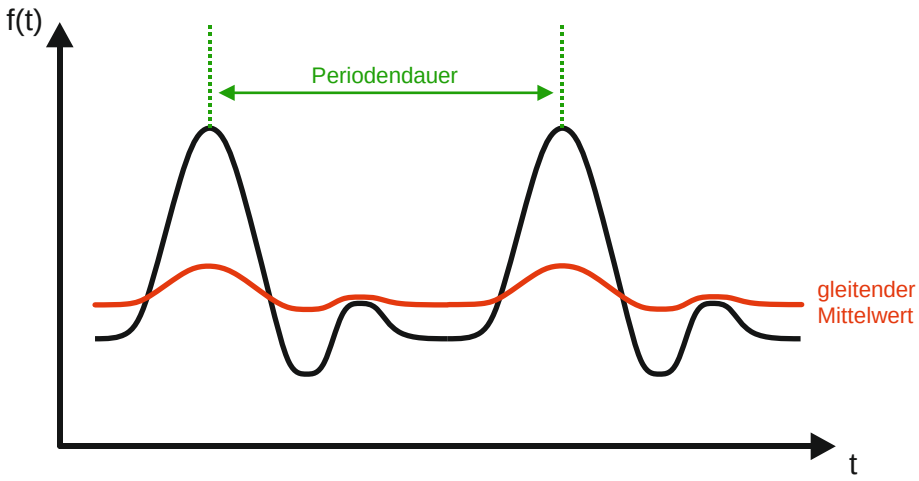

Abbildung 12: Gleitender Mittelwert zur Filterung des Gleichanteils im PPG-Signal

Diese Form der nachgeführten Mittelwertbildung wird als **gleitende Mittelwertbildung** bezeichnet. Abbildung 12 veranschaulicht diesen Sachverhalt. Zur programmtechnischen Umsetzung ist es erforderlich eine Anzahl n von Messwerten aus der Vergangenheit in einem Array zu speichern und aus diesen den Mittelwert zu bilden. Mit jedem neuen Messwert wird der jeweils älteste Messwert aus dem Array überschrieben und jedes Mal erneut der Mittelwert berechnet. Der Zugriff auf die Werte im Array muss nach Art eines Ring-Puffers erfolgen, d. h. es muss einen Schreibzeiger geben, der auf das nächste neu zu beschreibende Array-Element zeigt und beim Erreichen des letzten Array-Elementes als nächstes wieder ringförmig mit dem ersten Element fortfährt.

Die Mittelwertbildung über alle im Array gespeicherten Werte kann recheneffektiv in der Weise erfolgen, dass nicht jedes Mal alle Array-Elemente erneut summiert und dann die Summe durch n geteilt werden muss, sondern statt dessen nur der neue Messwert zu einer Summen-Variable hinzuaddiert wird und der jeweils älteste Wert des Arrays von dieser Variablen subtrahiert und diese dann durch n geteilt wird. Der Ablauf ist dann:

1. Warten bis ein neuer PPG-Messwert vorliegt
2. Dann den ältesten Wert des Arrays an der Schreibzeigerposition von der Summen-Variablen abziehen
3. Den neuen PPG-Messwert an dessen Stelle in das Array schreiben (überschreiben) …
4. … und zur Summenvariablen hinzuaddieren
5. Den Schreibzeiger ringförmig auf das nächste Array-Element ändern
6. Den Mittelwert durch Division der Summen-Variablen durch n berechnen

Hinweis: Wenn die Anzahl n der Werte über die ein Mittelwert gebildet werden soll so gewählt wird, dass sie einer Zweierpotenz entspricht und wenn die Summen-Variable die dividiert werden soll von einem vorzeichenlosen Integer-Typ ist, dann lässt sich die Divisionsoperation für die Mittelwertberechnung effektiv durch eine "*Shift*"-Operation ersetzen, für deren Umsetzung der Mikrocontroller nur einen Taktzyklus benötigt
(Beispiel: Division durch 8 entspricht "*Shift*"-Operation nach rechts um 3 Bitstellen).

4.1 Aufgabe: PPG-Sensor per I2C konfigurieren, auslesen und per USART als Textzeichen-Graph übertragen

In Teil 1 von Versuch 4 soll das an das Mikrocontroller-Board angeschlossene PPG-Modul unter Nutzung der zuvor beschriebenen Bibliotheksfunktionen für eine periodische Messung neuer PPG-Signalwerte konfiguriert und es sollen jeweils nach erfolgter Messung seine Messdaten periodisch ausgelesen und unter Zuhilfenahme des aus ihnen gebildeten gleitenden Mittelwerts gefiltert und skaliert werden. Der so entstandene Ergebniswert soll dann als Zeichenkette von m gleichen Zeichen je Zeile per USART 0 an den PC gesendet werden. Dabei soll die Anzahl m der je Zeile gesendeten Zeichen proportional zum gemessenen Ergebniswert sein. In der zeitlichen Folge mehrerer gesendeter Zeilen ergibt sich dann eine Balkendiagramm-ähnliche Darstellung eines PPG-Signalverlaufs-Graphen nach folgendem Beispiel:

Ergebniswert 50: ■■■■■■■■■
Ergebniswert 55: ■■■■■■■■■■
Ergebniswert 53: ■■■■■■■■■■
Ergebniswert 25: ■■■■■

Erstellen Sie hierzu ein neues C-Projekt mit dem Namen **SMRM_V4_Teil1** in *Atmel Studio*.

Anforderungen:

- Das Hauptprogramm *main* besteht aus einem Konfigurationsblock und einer Endlosschleife.
- Im Konfigurationsblock sollen die USART-Schnittstelle, der Port E, die I2C-Schnittstelle und das PPG-Modul passend konfiguriert und initialisiert werden.
- Die Zeichenübertragung der USART-Schnittstelle soll mit 19200 Baud erfolgen.
- Das PPG-Modul soll mit aktiviertem Interrupt im Herzraten-Modus betrieben, der integrierte FIFO-Puffer ringförmig bei 8-fachem *Oversampling* mit einer Auflösung von 18 Bit und einer Abtastrate von 400 sps beschrieben werden und die rote LED soll mit einem Strom von 6,2 mA betrieben werden.
- In der Endlosschleife des Hauptprogramms soll in jedem Durchlauf geprüft werden, ob neue Messdaten des PPG-Moduls vorliegen. Nur in dem Fall erfolgen alle nachfolgenden Schritte.
- Der neue Messwert soll gelesen und von den 18 Bit sollen die niederwertigen 16 Bit (Typ *uint16_t*) weiter genutzt werden.
- Der gleitende Mittelwert aus 128 Messwerten soll auf Basis des neuen Wertes aktualisiert und in der im Text beschriebenen Weise zur Filterung des Wechselanteils des PPG-Signals genutzt werden (Typ *int16_t*).
- Der so entstandene Wechselanteil, der sowohl positive, als auch negative Werte annehmen kann, ist durch geeignete Skalierung (Faktor und Offset) in eine positive (vorzeichenlose) Zahl m im Wertebereich von 0 bis 31 zu überführen (Typ *uint8_t*; brauchbare initiale Startwerte sind Faktor = 1/64 und Offset = 20, können aber modifiziert werden).
- Der so skalierte Wert m gibt die Anzahl von gleichen Symbolzeichen an, die abschließend als Textzeichenfolge einen Balken mit wertproportionaler Länge bilden und über den USART0 mit jedem Messzyklus an den PC übertragen und dort über das Terminalprogramm empfangen und visualisiert werden sollen.

4.2 Vorbereitungsaufgaben

Für die Beantwortung der nachfolgenden Fragen finden Sie alle notwendigen Informationen in den Unterlagen der Praktikumsversuche 1 bis 4, im Datenblatt des ATmega328PB und im Datenblatt des MAX30102.

- Wie ist der Port E des ATmega328PB für die Verwendung mit dem angeschlossenen MAX30102 nach den Vorgaben im Abschnitt 4 initial zu konfigurieren?

  DDRE = _____ PORTE = _____

- Mit welcher Periodendauer werden neue Messwerte vom MAX30102 zur Verfügung gestellt, wenn dieser, wie angegeben, mit einer Messrate von 400 sps und 8-fachem *Oversampling* betrieben wird?

  _____ ms

- Mittels welcher C-Befehlszeile werden die niederwertigen 16 Bit des 18-Bit PPG-Messwerts, der als 3-Byte-Zahl aus dem FIFO-Puffer des MAX30102 ausgelesen und Byte-weise in das Array *buffer* geschrieben wurde, aus diesem ausgelesen und stellenrichtig in die Variable *ppg_raw* (Typ *uint16_t*) kopiert?

  ppg_raw = _____

- Wie wird ein Array *buffer* für einen Ringpuffer für 128 vorzeichenlose ADU-Werte mit je 16 Bit Länge in C deklariert?

  _____

- Der Ringpuffer soll mit einer Zählvariablen *buffer_index* durchlaufen, bzw. die aktuelle Position angezeigt werden. Mit welchem arithmetischen Operator kann in C sichergestellt werden, dass nach Erreichen des letzten Elements im Array *buffer* wieder mit dem ersten Element begonnen wird? Schreiben sie die vollständige C-Befehlszeile.

  _____

- Wie kann in C in der Interruptserviceroutine des Timers ein Ringpuffer mit dem neuen PPG-Wert *ppg_raw* aktualisiert und mit diesem der gleitende Mittelwert *ppg_avg* gebildet werden? Nutzen sie die Summenvariable *buffer_sum* und vermeiden sie Schleifen.

  _____

  _____

  _____

  _____

  _____

5 Algorithmus zur Berechnung der Herzfrequenz aus einem Signalverlauf

Im vorangegangenen Teilversuch 1 wurde der aktuelle Messwert des PPG-Moduls periodisch über den I2C-Bus ausgelesen und als Balkendiagramm dargestellt. Diese Anzeige gibt näherungsweise die schnellen Blutflussänderungen am Ort der Messung direkt wieder, die durch den Herzschlag erzeugt werden.
Im nächsten Schritt soll aus dem Verlauf der Messwerte der zeitliche Abstand von einem zum nächsten Pulsschlag ermittelt, und damit die aktuelle Herzfrequenz in Schlägen pro Minute (bpm, *beats per minute*) berechnet werden. Um dieses Ziel zu erreichen müssen zwei Aufgaben bewältigt werden: Der Algorithmus zur Berechnung der Herzfrequenz muss in der Lage sein die Pulsschläge im Signalverlauf zu erkennen und von anderen Signaländerungen zu unterscheiden und er muss über eine präzise Methode zur Erfassung des zeitlichen Abstands zwischen den Pulsschlägen verfügen.
Die beste Möglichkeit zur präzisen Zeitmessung mit einem Mikrocontroller stellt die Verwendung von Timern dar. Solange die Arbeitsfrequenz eines Timers von einem Schwingquarz abgeleitet wird, lassen sich Ereignisse im Programmablauf mit genau definiertem zeitlichem Abstand wiederholen, sodass in Verbindung mit einer Zählvariablen auf einfache Art eine genaue Zeitmessung im Raster der Timerintervalle möglich ist.
Für die Erkennung der Pulsschläge aus dem Verlauf der Messwerte soll im Folgenden ein Algorithmus vorgestellt und im Praktikum implementiert werden.

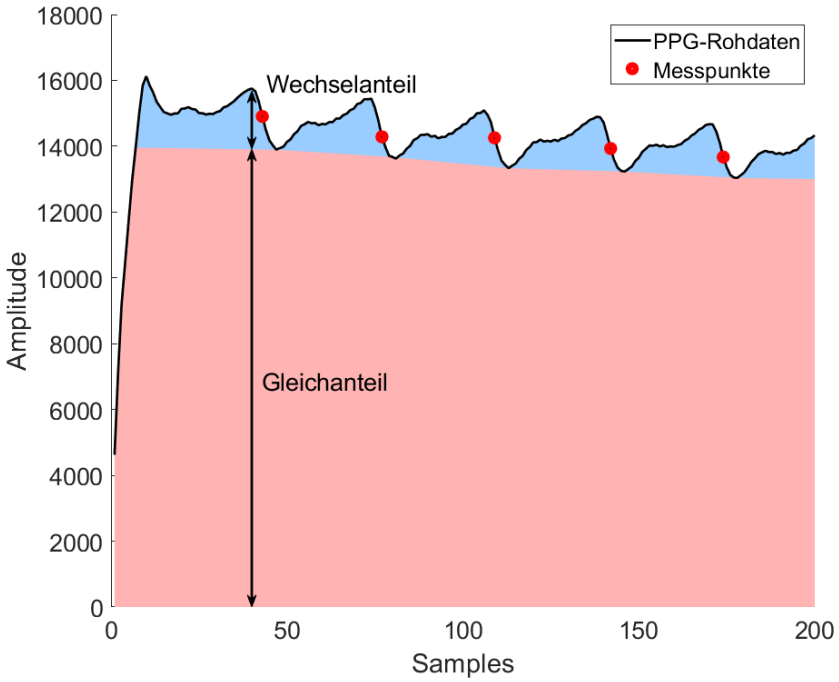

Abbildung 13: Unbearbeitetes PPG-Signal des Sensors

Der beschriebene Algorithmus besteht aus zwei Teilen, der Aufbereitung (Filterung) der einkommenden Daten und der anschließenden Merkmalsextraktion und Messung der Periodendauer.
Das vom Sensor erfasste und digitalisierte PPG-Signal besteht, wie schon in Teil 1 beschrieben, aus einem kleinen Wechselanteil, der die Pulsinformation enthält, und einem

großen Gleichanteil, der keine Bedeutung für den Algorithmus hat (siehe Abbildung 13). Daher erfolgt zu Beginn eine Reduktion der Datenwortbreite, wobei durch geschickte Wahl der Bitstellen der Wechselanteil nahezu vollständig vom Gleichanteil getrennt wird. Zu diesem Zweck werden vom 18 Bit breiten Ausgangssignal [17:0] nur 8 Bit zur weiteren Verarbeitung verwendet [12:5]. Das daraus resultierende Signal ist in Abbildung 14 zu sehen und reduziert auf der einen Seite den Rechenaufwand der folgenden Schritte, ist auf der anderen Seite aber auch detailliert genug für die Ermittlung der Herzfrequenz.

Zur Messung der Periodendauer aus einem Signalverlauf muss ein Merkmal des Signals genutzt werden, welches sich nicht nur periodisch stabil wiederholt, sondern sich auch gut extrahieren lässt. Hierzu wird der Zeitpunkt der betragsmäßig größten Steigung des Signals auf der fallenden Flanke zu Beginn einer Periode genutzt. Dieser Punkt korrespondiert mit dem Anstieg der Blutmenge im Gewebe unter dem Sensor, der beim Eintreffen der Pulswelle an diesem Ort auftritt, und ist damit direkt auf die Herzaktivität zurückzuführen. Die Berechnung der Steigung erfolgt als zeitliche Ableitung indem jeweils die Differenz zwischen dem aktuellen und dem vorangegangenen Messwert gebildet wird (siehe Abbildung 15).

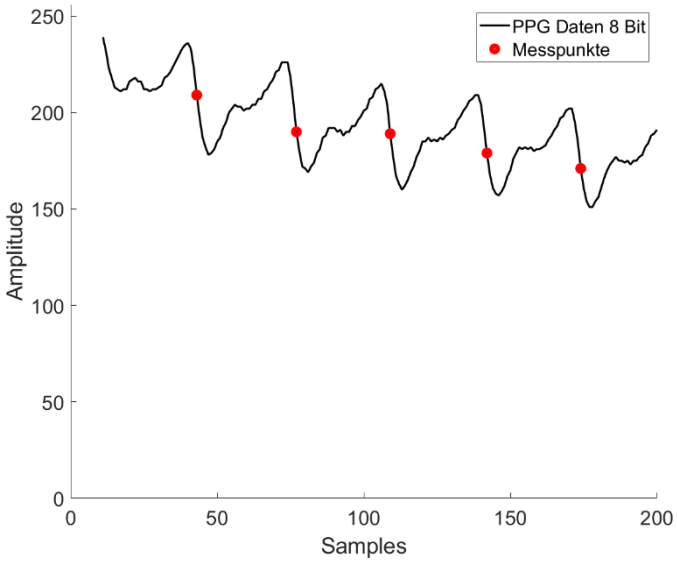

Abbildung 14: 8 Bit Zuschnitt der PPG-Daten

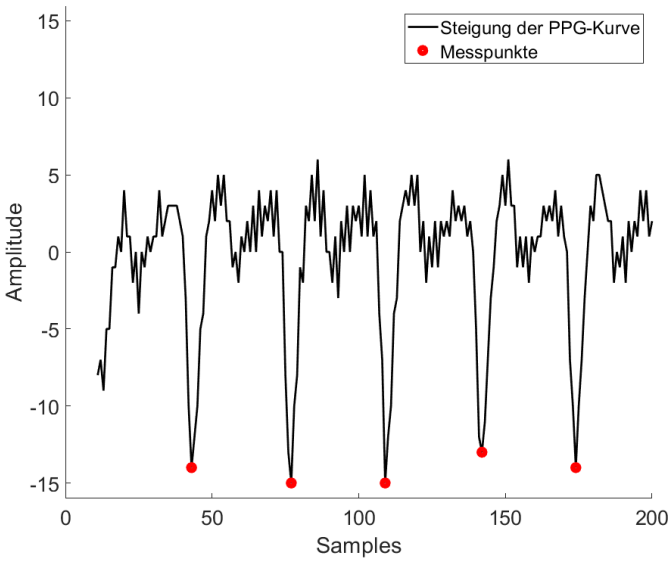

Abbildung 15: Zeitliche Ableitung des PPG-Signals

Durch die darauffolgende Quadrierung der Steigungswerte heben sich die gesuchten Punkte noch stärker gegenüber dem restlichen Signalverlauf hervor und die weitere Verarbeitung wird insofern vereinfacht, als dass nur noch positive Werte vorliegen. Weiter herausgearbeitet werden die gesuchten Signalspitzen, indem alle Werte unterhalb einer Grenze von ca. 100 auf 0 gesetzt werden (siehe Abbildung 16). Mit diesem Schritt ist die Aufbereitung des Signals abgeschlossen.

Zur Messung der Periodendauer muss jeweils der genaue Zeitpunkt des lokalen Maximums innerhalb der Signalspitzen ermittelt werden. Zu diesem Zweck wird der Signalverlauf durch ein "Fenster" betrachtet, das jeweils 3 Werte umfasst, wobei der jeweils aktuelle Wert mit den beiden vorangegangenen Werte verglichen wird (siehe Abbildung 17). Wenn nun der mittlere Wert im Fenster sowohl größer als der rechte als auch größer oder gleich dem linken Wert ist, stellt der mittlere Wert das lokale Maximum dar. Zur Ermittlung der Periodendauer zwischen zwei lokalen Maxima wird ein Zähler genutzt, der mit jedem neuen Messwert inkrementiert und nach Erreichen eines Maximus zurückgesetzt wird. Somit ist der Wert im Zähler bei Detektion eines Maximus proportional zum zeitlichen Abstand zum vorherigen Maximum. Die Herzfrequenz in bpm wird durch die Division der Abtastungen pro Minute durch die Zeitdifferenz der Maxima ermittelt. Um Fehler durch hochfrequente Störungen im Signal zu unterdrücken, werden Maxima, die zu einer Periodendauer unterhalb eines Zählerwerts von 15 führen (→ 200 bpm), ignoriert.

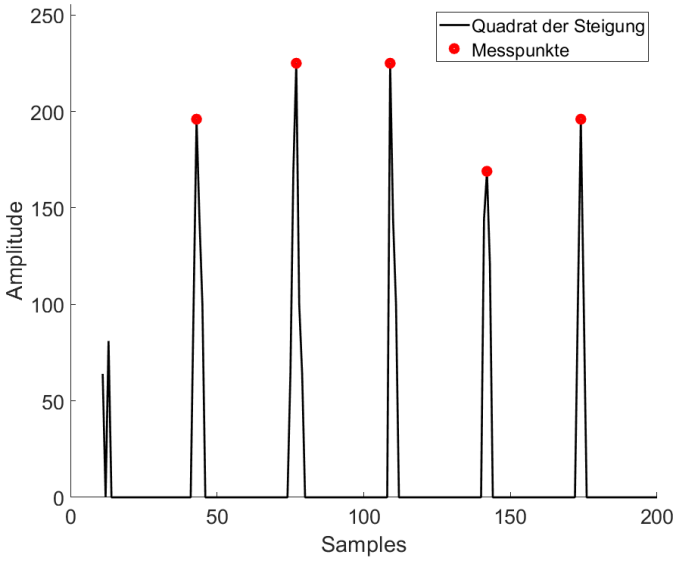

Abbildung 16: Quadrat der Ableitung unter Berücksichtigung des Grenzwerts

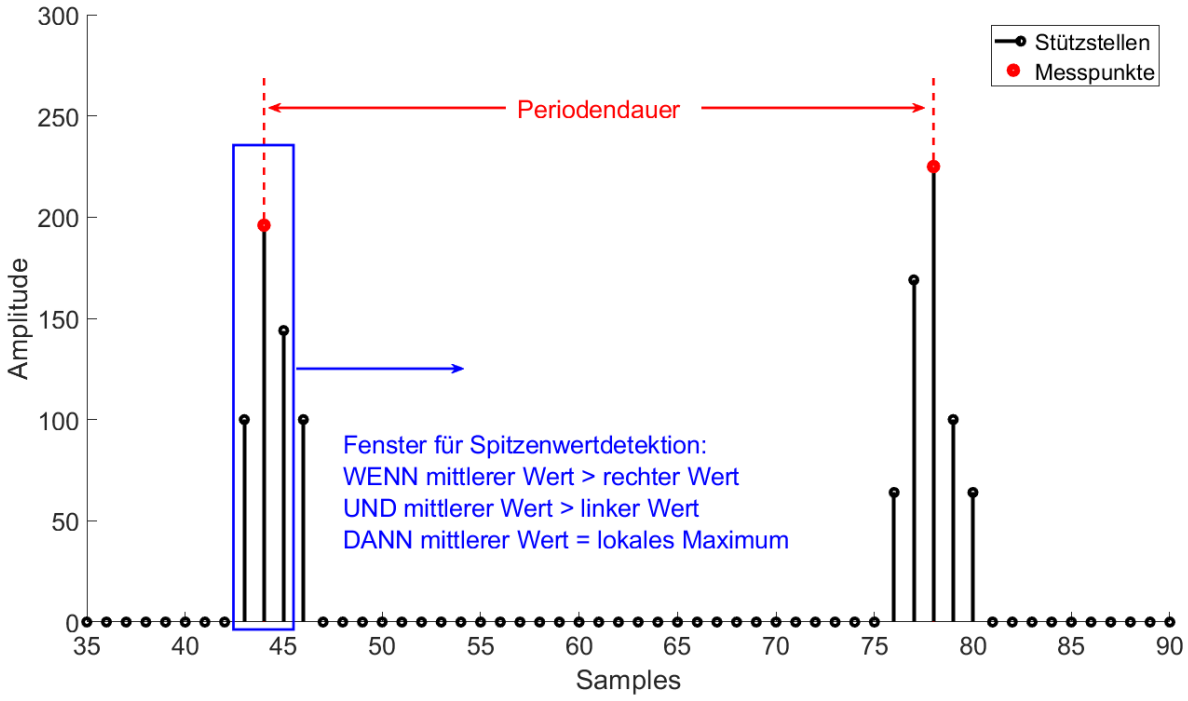

Abbildung 17: Detektion der lokalen Maxima zur Berechnung der Periodendauer

6 Praktischer Teil zum Versuch 4 Teil 2

6.1 Aufgabe: Herzfrequenz aus PPG-Wertefolge berechnen und per USART als Textzeichenfolge übertragen

In Teilversuch 2 des Versuchs 4 soll, wie schon in Teilversuch 1, das an das Mikrocontroller-Board angeschlossene PPG-Modul unter Nutzung der beschriebenen Bibliotheksfunktionen für eine periodische Messung neuer PPG-Signalwerte konfiguriert und es sollen jeweils nach erfolgter Messung seine Messdaten periodisch ausgelesen und über mehrere Messzyklen hinweg gemäß dem in Abschnitt 5 beschriebene Algorithmus zur Berechnung eines Herzfrequenz-Wertes genutzt werden. Der so entstandene Ergebniswert soll dann, mit jedem neuen erkannten Herzschlag als Zeichenkette des Zahlenwerts und seiner Einheit per USART 0 an den PC gesendet werden. Erstellen Sie hierzu ein neues C-Projekt mit dem Namen **SMRM_V4_Teil2** in *Atmel Studio*.

Anforderungen:

- Das Hauptprogramm *main* besteht aus einem Konfigurationsblock und einer Endlosschleife.
- Im Konfigurationsblock sollen die USART-Schnittstelle, der Port E, die I2C-Schnittstelle und das PPG-Modul passend konfiguriert und initialisiert werden.
- Die Zeichenübertragung der USART-Schnittstelle soll mit 19200 Baud erfolgen.
- Das PPG-Modul soll mit aktiviertem Interrupt im Herzraten-Modus betrieben, der integrierte FIFO-Puffer ringförmig bei 8-fachem *Oversampling* mit einer Auflösung von 18 Bit und einer Abtastrate von 400 sps beschrieben werden und die rote LED soll mit einem Strom von 6,2 mA betrieben werden.
- In der Endlosschleife des Hauptprogramms soll in jedem Durchlauf geprüft werden, ob neue Messdaten des PPG-Moduls vorliegen. Nur in dem Fall erfolgen alle nachfolgenden Schritte.
- Der neue Messwert soll gelesen und von den 18 Bit sollen die Bits 5 bis 12 als vorzeichenloses 8-Bit-Datenwort (Typ *uint8_t*) weiter genutzt werden.
- Von dem so gefilterten Datenwort soll dann die Differenz zum Datenwort der vorherigen Messung gebildet werden (diskrete Ableitung). Anschließend ist der aktuelle Messwert als neuer Vorgängerwert für die nächste Differenzenbildung zu speichern.
- Der neue Ableitungswert ist anschließend zu quadrieren ("verstärkt" große Werte) und mit einem Schwellwert (z. B. 100) zu filtern. Ergebnis ist eine positive Zahl ≥100.
- Diese Quadratzahl wird mit ihren beiden Vorgängern einer Mustersuche unterzogen (Folge mit Größtwert in der Mitte) und bei einem Treffer entspricht der aktuelle Wert eines Zykluszählers dem zeitlichen Pulsabstand. Der Zykluszähler wird dann zurückgesetzt, ansonsten Inkrementiert. Anschließend ist der aktuelle Wert der Quadratzahl als neuer Vorgängerwert für die nächste Spitzenwertsuche zu speichern.
- Aus gefundenen Pulsabstand-Zählwerten ist dann die Herzfrequenz zu berechnen.
- Mit jedem neuen Herzfrequenz-Wert ist dieser über den USART0 als Zeichenkette aus seinem Zahlenwert und seiner Einheit an den PC zu übertragen.

6.2 Vorbereitungsaufgaben

Für die Beantwortung der nachfolgenden Fragen finden Sie alle notwendigen Informationen in den Unterlagen der Praktikumsversuche 1 bis 4, im Datenblatt des ATmega328PB und im Datenblatt des MAX30102.

- Mittels welcher C-Befehlszeile werden die Bits 5 bis 12 des 18-Bit PPG-Messwerts, der als 3-Byte-Zahl aus dem FIFO-Puffer des MAX30102 ausgelesen und Byte-weise in das Array *buffer* geschrieben wurde, stellenrichtig in die Variable *ppg_raw* (Typ *uint8_t*) kopiert?

  ppg_raw = _____

- Mittels welcher beiden C-Befehlszeilen wird ein Zahlenwert in der Variablen *ppg_gradient* quadriert und nur Quadrat-Werte ≥100 direkt, andernfalls der Wert 0 einer Variablen *peaks* zugewiesen?

  _____

  _____

- Mit welcher C-Anweisung wird eine Dreierfolge von Werten einer Mustersuche unterzogen, wobei der mittlere Wert (*peaks_history*[0]) der Größte sein muss, der erste (älteste) Wert (*peaks_history*[1]) gleich oder kleiner ist und der letzte (neueste) Wert (*peaks*) kleiner als die anderen ist?

  if (_____) {}

- Mit welcher C-Anweisung wird bei einer Messwiederholrate von 50 Hz aus dem Wert einer Zählvariablen *peaks_delta_t*, deren Inhalt der Anzahl von Messzyklen zwischen zwei Pulsschlägen entspricht, die Herzfrequenz in Schlägen pro Minute berechnet und in der Variablen *heartrate* gespeichert?

  _____

- Welche C-Anweisung überträgt den Wert der berechneten Variablen *heartrate* über den USART0 an den PC so, dass dieser als Textzeichenfolge aus Dezimalzahl mit Einheit jeweils in einer eigenen Zeile ausgegeben wird?

  _____

Anhang A: Konfigurationsregister des PPG-Moduls MAX30102

Registertabelle und Kurzbeschreibungen Teil1:

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
STATUS											
Interrupt Status 1	A_FULL	PPG_ RDY	ALC_ OVF					PWR_ RDY	0x00	0X00	R
Interrupt Status 2							DIE_TEMP _RDY		0x01	0x00	R
Interrupt Enable 1	A_FULL_ EN	PPG_ RDY_EN	ALC_ OVF_EN						0x02	0X00	R/W
Interrupt Enable 2							DIE_TEMP _RDY_EN		0x03	0x00	R/W
FIFO											
FIFO Write Pointer				FIFO_WR_PTR[4:0]					0x04	0x00	R/W
Overflow Counter				OVF_COUNTER[4:0]					0x05	0x00	R/W
FIFO Read Pointer				FIFO_RD_PTR[4:0]					0x06	0x00	R/W
FIFO Data Register	FIFO_DATA[7:0]								0x07	0x00	R/W
CONFIGURATION											
FIFO Configuration	SMP_AVE[2:0]			FIFO_ ROLL OVER_EN	FIFO_A_FULL[3:0]				0x08	0x00	R/W
Mode Configuration	SHDN	RESET				MODE[2:0]			0x09	0x00	R/W
SpO_2 Configuration	0 (Reserved)	SPO2_ADC_RGE [1:0]		SPO2_SR[2:0]			LED_PW[1:0]		0x0A	0x00	R/W
RESERVED									0x0B	0x00	R/W
LED Pulse Amplitude	LED1_PA[7:0]								0x0C	0x00	R/W
	LED2_PA[7:0]								0x0D	0x00	R/W
RESERVED									0x0E	0x00	R/W
RESERVED									0x0F	0x00	R/W
Multi-LED Mode Control Registers		SLOT2[2:0]				SLOT1[2:0]			0x11	0x00	R/W
		SLOT4[2:0]				SLOT3[2:0]			0x12	0x00	R/W

Registertabelle und Kurzbeschreibungen Teil2:

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
RESERVED									0x13–0x17	0xFF	R/W
RESERVED									0x18-0x1E	0x00	R
DIE TEMPERATURE											
Die Temp Integer	TINT[7:0]								0x1F	0x00	R
Die Temp Fraction					TFRAC[3:0]				0x20	0x00	R
Die Temperature Config								TEMP_EN	0x21	0x00	R/W
RESERVED									0x22–0x2F	0x00	R/W
PART ID											
Revision ID	REV_ID[7:0]								0xFE	0xXX*	R
Part ID	PART_ID[7]								0xFF	0x15	R

**XX denotes a 2-digit hexadecimal number (00 to FF) for part revision identification. Contact Maxim Integrated for the revision ID number assigned for your product.*