

HÖHERE TECHNISCHE BUNDESLEHRANSTALT HOLLABRUNN

Höhere Abteilung für Elektronik – Technische Informatik

Klasse/ Jahrgang:	Übungsbetreuer:
4BHEL	Dipl.-Ing. Josef Reisinger
Übungsnummer:	Übungstitel:
4	Gestensensor APDS-9960
Datum der Vorführung:	Gruppe:
17.06.2020	Clemens Marx, Stefan Grubmüller
Datum der Abgabe:	Unterschrift:
17.06.2020	

Beurteilungskriterien

Programm:	Punkte
Programm Demonstration	
Erklärung Programmfunktionalität	
Protokoll:	Punkte
Pflichtenheft (Beschreibung Aufgabenstellung)	
Beschreibung SW Design (Flussdiagramm, Blockschaltbild,..)	
Dokumentation Programmcode	
Testplan (Beschreibung Testfälle)	
Kommentare / Bemerkungen	
Summe Punkte	

Note: _____

Gestensensor APDS-9960



Inhaltsverzeichnis

1	Originalangabe und Unterschriften	3
2	Gestensensor APDS9960	4
2.1	Allgemeines.....	4
2.2	Blockdiagram	4
2.3	Pin-Konfiguration.....	5
2.4	Blockschaltbilder	5
2.5	Elektrische Charakteristiken	7
2.6	I ² C-Bus Protokoll.....	7
2.6.1	Bus Zustände.....	7
2.6.2	Write Befehl	8
2.6.3	Read Befehl	8
2.7	Systemtakt HSE 36MHz.....	9
2.8	Uhr.....	10
2.9	Gestensensor.....	10
2.10	Abstandssensor	10
2.10.1	Register.....	11
2.10.2	Enable Register (0x80).....	11
2.10.3	Proximity Interrupt Threshold Register (0x89/0x8B)	12
2.10.4	Persistence Register (0x8C).....	12
2.10.5	Proximity Pulse Count Register (0x8E).....	12
2.10.6	Control Register One (0x8F)	13
2.10.7	Configuration Register Two (0x90)	13
2.10.8	Status Register (0x93).....	14
2.10.9	Proximity Data Register (0x9C)	14
2.10.10	Configuration Three Register (0x9F).....	14
2.11	Testprogramm.....	15
2.11.1	Konfiguration.....	15
2.11.2	Externer Interrupt PA1.....	17
2.12	Probleme.....	18
2.13	Erkenntnisse	18
2.14	Zeitaufwand – Stefan Grubmüller.....	18
2.15	Zeitaufwand – Clemens Marx.....	18

Anhang:

- Source-Code
- Datenblatt

1 Originalangabe und Unterschriften

Digitale Systeme 4.Jg Übung **ARM-Interrupt** Angabe Nr: **01**

Betreuer: REJ

Übungstitel: **Gestensensor APDS-9960**

Übungsdatum:

Hardwarekomponente: Sparkfun APDS-9960 Sensor Modul

Uhr mit Timer: Timer 3

Zu verwendender Systemtakt: HSE 36Mhz

Entwickle in der Programmiersprache C eine interrupt gesteuerte Software für den Cortex-M3 Mikrocontroller welche mehrere parallele Prozesse realisiert. (Echtzeitsystem)
Folgende Struktur soll dabei realisiert werden

Prozess A: Hauptprg, Initialisierung, Visualisierung von Messwerten und Uhr auf Anzeige**Prozess B:** Uhrenfunktion (gemeinsamer Speicher mit Prozess A)**Prozess C:** Erfassung von Messdaten (z.B.: Drehzahlmessung)

Am Beginn des Programms soll zunächst eine **Begrüßungstext** ausgegeben werden, der zu einem Hardwaretest auffordert (Komponente vorhanden oder nicht vorhanden)

Allgemeine Regeln für diesen Übungsdurchgang:

- Es ist ein detailliertes Pflichtenheft zu erarbeiten
- Zuerst ist als Programmbeschreibung ein **Blockschaltbild** zu zeichnen, welches **alle** Information in kommentierte graphischer Form darstellt.
- Timer, Counter Konstanten sind zu berechnen und graphisch darzustellen
- Uhrzeit ist im ASCII Code in folgender Form anzuzeigen **hh:mm:ss:z**
- Vorsicht: print sind nicht reentrant !! und soll deshalb in keiner ISR verwendet werden
- Der Source Code ist entsprechend zu **dokumentieren** bzw. soll es für jede Funktion einen entsprechenden **Funktionskopf** gegen der Funktion beschreibt (Aufgabe der Funktion, Input bzw. Output Parameter und eventuelle Error Codes) – idealerweise nach doxygen Standard
- Sinnvoll ist für die Fehlersuche bzw. anschließend für den Funktionsnachweis Oszilloskop bzw. Logikanalysator eventuell einzusetzen

Angabebesprechung, Schriftliches, detailliertes Pflichtenheft (incl. Bitbelegung und Timing)	
Blockschaltbild	
Vorführung des lauffähigen Programms, Testdatensatz und Demonstration der Tabelle ist Bestandteil der Vorführung	
Protokollabgabe	

Protokollaufbau (Reihenfolge und Nummerierung einhalten!!!!):

- 1.) Inhaltsverzeichnis
- 2.) Originalangabe und Unterschriften
- 3.) Pflichtenheft (Angabekonkretisierung, -erweiterung, -einschränkung, -änderung),
Bitbelegung
- 4.) Algorithmusbeschreibung: Beschreibung des Gesamtsystems mithilfe eines Blockschaltbildes , ev. mit Foto
- 5.) Kommentiertes Listing bzw. Header Datei der Library (*.h)
- 6.) Testplan (Nachweis der einzelnen Teilfunktionen, Wirkung extremer Eingaben, Grenzfälle, wann kommt's zum Absturz ...)
- 7.) Zeitaufwand (aufgeschlüsselt) und Arbeitsteilung
- 8.) Aufgetretene Probleme
- 9.) Betrachtung der Ergebnisse und Erkenntnisse aus der Übung

2 Gestensensor APDS9960

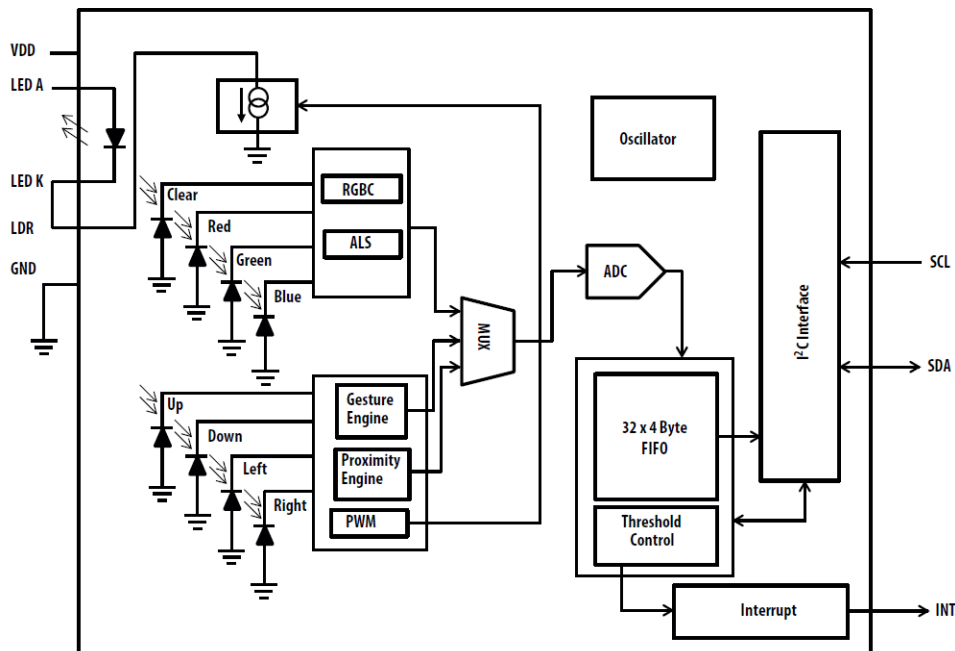
2.1 Allgemeines

Das Sensor APDS9960 unterstützt Funktionen wie Gestenerkennung, Abstandserkennung, Umgebungslicht und Farben im RGBC Spektrum. Die gesamte Platine umfasst eine Fläche von rund 9,3mm². Für die Gestenerkennung gibt es vier Fotodioden, welche Infrarotlicht von einer integrierten LED, welche von Objekten, wie zum Beispiel einer Hand reflektiert werden, erkennen und in ein digitales Signal umwandeln. Erkannt werden können einfache UP-DOWN-LEFT-RIGHT Bewegungen sowie kompliziertere Bewegungen (diagonal, ...). Stromverbrauch wird bei diesem Sensor durch Sleep Mode und einstellbaren Infrarot und LED Timing minimiert. Der Abstandssensor bietet Distanzmessung (z.B.: Mobilgerät zum Ohr beim Telefonieren). Die Erkennung erfolgt mittels Interrupts, welche immer dann ausgelöst werden, wenn das Messergebnis innerhalb der unteren und oberen Grenze liegt. Der Sensor kann auch mittels der Fotodioden für Rot, Grün, Blau und normales Licht die Intensität herausfinden. Damit kann genau das Umgebungslicht und Farben gemessen werden, was Geräten ermöglicht die Farbtemperatur zu berechnen oder das Hintergrundlicht bei Displays zu steuern. Die Kommunikation mit dem Sensor erfolgt bei uns mittels I²C-Bus Protokoll. Als I²C-Library wurde die von Jakob Pachtrog verwendet.



2.2 Blockdiagramm

Functional Block Diagram



Blockdiagramm des APDS9960

Vereinfacht ausgedrückt, werden über die Fotodioden Messwerte aufgenommen, welche über den Speicher mit FIFO Prinzip („First in, first out“ – „Erster rein, erster raus“), welcher über den I²C-Bus angesteuert werden, ausgewertet. Dementsprechend löst dieser dann über den Interrupt Pin ein Interrupt Signal aus. Dieses Interrupt Signal wird bei uns beim Cortex über einen Externen Interrupt über den Pin PA7 geregelt.

2.3 Pin-Konfiguration

I/O Pins Configuration

Pin	Name	Type	Description
1	SDA	I/O	I ² C serial data I/O terminal - serial data I/O for I ² C-bus
2	INT	O	Interrupt - open drain (active low)
3	LDR		LED driver input for proximity IR LED, constant current source LED driver
4	LEDK		LED Cathode, connect to LDR pin when using internal LED driver circuit
5	LEDA		LED Anode, connect to V _{LEDA} on PCB
6	GND		Power supply ground. All voltages are referenced to GND
7	SCL	I	I ² C serial clock input terminal - clock signal for I ² C serial data
8	V _{DD}		Power supply voltage

Für unserer Anwendung werden nur fünf Anschlüsse verwendet. SDA (Signal Data) und SCL (Signal Clock) sind die Ansteuerleitungen für den I²C-Bus. Über die SDA-Leitung werden die Daten gesendet und über die SCL-Leitung das Clock Signal. Die Daten auf der Datenleitung können sich bidirektional bewegen (Input und Output). Zur Versorgung verwenden wir über den V_{DD} Pin den 3,3V Anschluss auf unserem Mikrocontroller sowie Masse über den GND Pin. Der Interrupt Pin wird verwendet, um eine Ausgabe bei Erreichen entsprechender Messwerte auszugeben und am Cortex über den Pin PA7 über einen Externen Interrupt aufzunehmen.

2.4 Blockschaltbilder

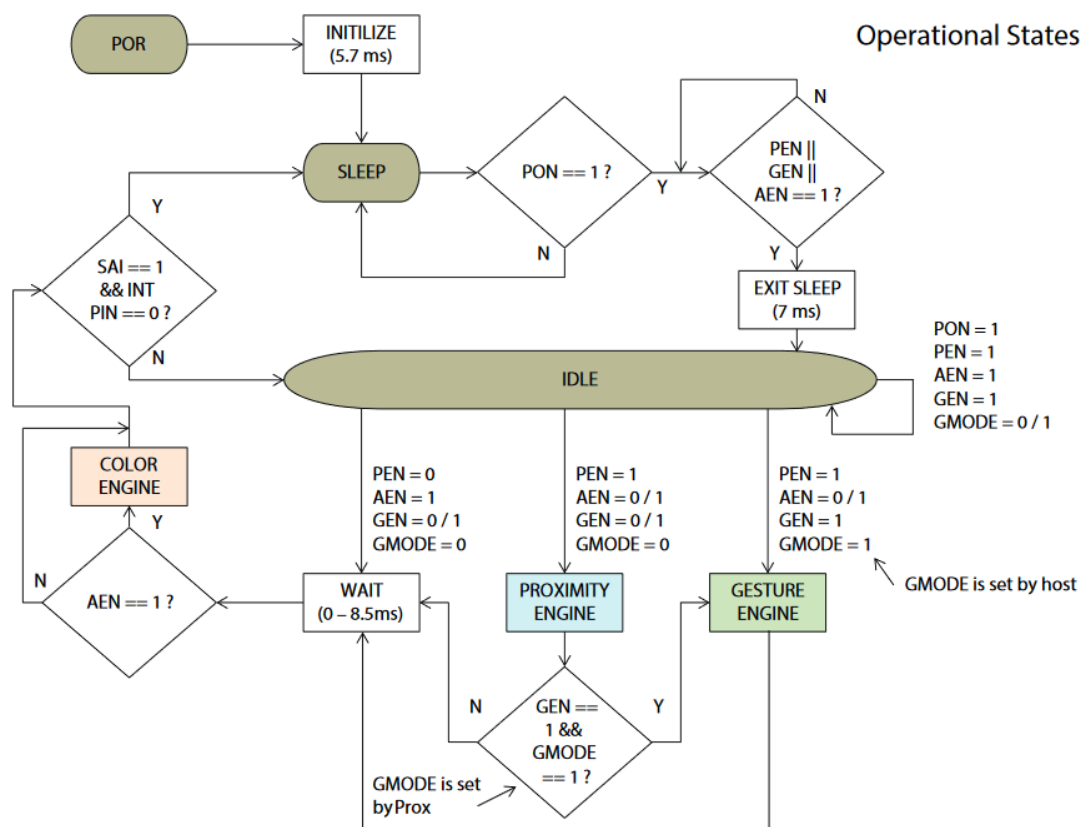
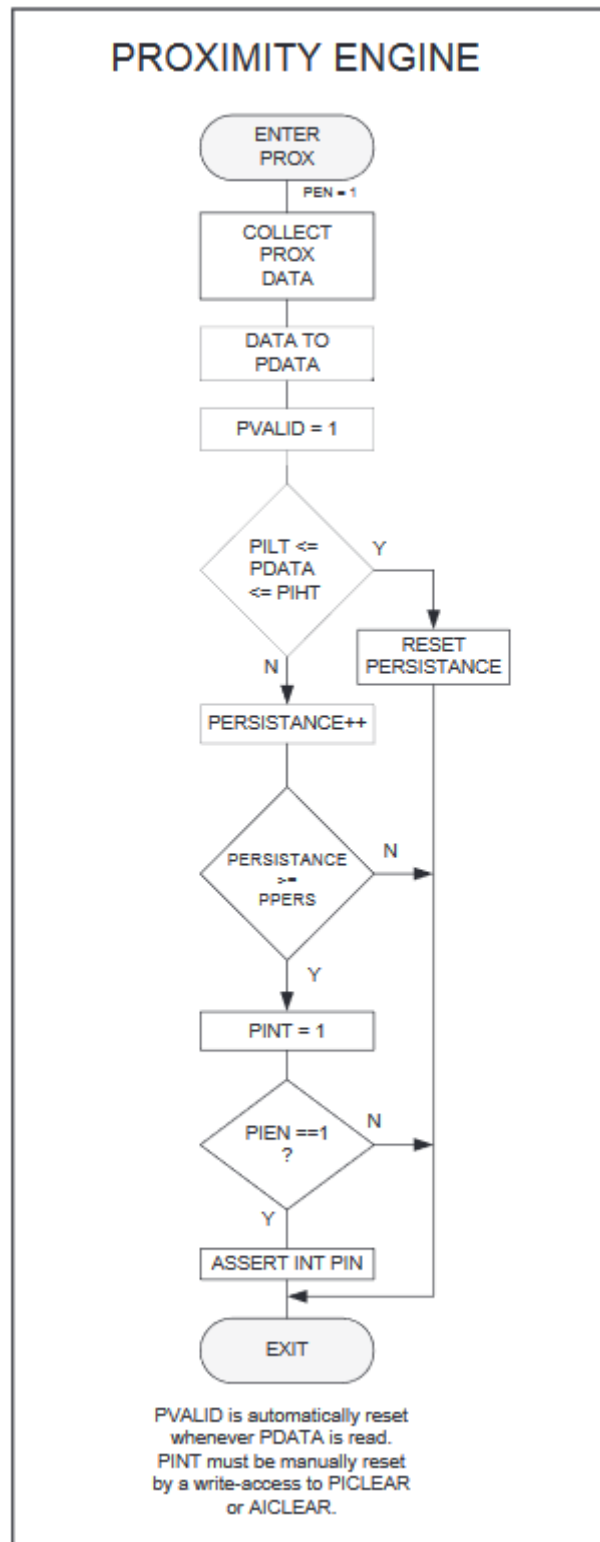


Figure 7. Detailed State Diagram



Auf diesen Blockschaltbildern erkennt man die genaue Funktionsweise des Sensors, welche Bits gesetzt und geprüft werden.

2.5 Elektrische Charakteristiken

AC Electrical Characteristics, $V_{DD} = 3\text{ V}$, $T_A = 25\text{ }^{\circ}\text{C}$ (unless otherwise noted) *

Parameter	Symbol	Min.	Max.	Unit
Clock frequency (I ² C-bus only)	f_{SCL}	0	400	kHz
Bus free time between a STOP and START condition	t_{BUF}	1.3	–	μs
Hold time (repeated) START condition. After this period, the first clock pulse is generated	t_{HDSTA}	0.6	–	μs
Set-up time for a repeated START condition	$t_{SU,STA}$	0.6	–	μs
Set-up time for STOP condition	$t_{SU,STO}$	0.6	–	μs
Data hold time	$t_{HD,DAT}$	30	–	ns
Data set-up time	$t_{SU,DAT}$	100	–	ns
LOW period of the SCL clock	t_{LOW}	1.3	–	μs
HIGH period of the SCL clock	t_{HIGH}	0.6	–	μs
Clock/data fall time	t_f	20	300	ns
Clock/data rise time	t_r	20	300	ns
Input pin capacitance	C_i	–	10	pF

Als Clock Frequenz wurde der Einfachheit halber 80kHz genommen. Darüber wäre der I²C-Bus nämlich nicht mehr im Standard Mode, sondern im Fast Mode (100 bis 400kHz) und es müssten mehr Konfigurationen getroffen werden (Low/High Periode von der Clock, ...).

2.6 I²C-Bus Protokoll

Der I²C Bus ist ein synchroner serieller 2-Draht Bus, der in den 80er Jahren von Philips entwickelt wurde. I²C gesprochen 'I Quadrat C' kommt von der der Abkürzung IIC und bedeutet Inter-Integrated Circuit. Er wird hauptsächlich dazu benutzt, zwischen Schaltkreisen, die sich auf einer Platine verbinden, Daten auszutauschen. Die beiden Leitungen, die den I²C Bus bilden heißen SCL und SDA. SCL steht für Signal Clock und ist die Taktleitung für den Bus. Deshalb spricht man auch von einem synchronen Bus. SDA steht für Signal Data und ist die Datenleitung. Die Datenübertragungsrate des I²C Busses beträgt 100kHz im Standard Mode, bzw. 400kHz im Fast Mode.

Der I²C Bus ist ein Multi Master/Slave Bus. Das bedeutet, es gibt mindestens einen I²C Master und ebenso mindestens einen I²C Slave. Der Master selektiert einen Slave durch seine Slave Adresse, die innerhalb eines Busses eindeutig sein muss. Eine Datenübertragung kann nur durch einen I²C Master initiiert werden. Der Slave bleibt immer passiv und lauscht nur auf die Slave Adresse und vergleicht diese mit seiner eigenen Slave Adresse. Erst wenn er seine Slave Adresse erkennt, greift der Slave auch aktiv in das Busgeschehen ein.

Aus Sicht des I²C Masters unterscheidet man zwischen Read und Write Sequenzen. Bei einer Read Sequenz liest der I²C Master Daten vom I²C Slave. Bei einer Write Sequenz sendet der I²C Master Daten zum Slave.

2.6.1 Bus Zustände

Bus frei: Wenn SCL und SDA dauerhaft HIGH sind, spricht man von 'Bus free'. Ein I²C Master muss diese Bedingung immer zuerst abprüfen, bevor er den Bus belegen darf.

Start Bedingung: Die Start Bedingung kennzeichnet den Beginn einer Datenübertragung durch einen I²C Master. Der Master zieht die Datenleitung SDA von HIGH auf LOW, während die Taktleitung SCL auf HIGH bleibt.

Datenbit: Ein Datenbit kann, wie in der Digitaltechnik üblich 2 Zustände einnehmen '0' oder '1'. Die Daten sind gültig während die Taktleitung SCL auf HIGH liegt. Ein LOW Pegel auf der Datenleitung SDA bedeutet '0', ein HIGH bedeutet '1'.

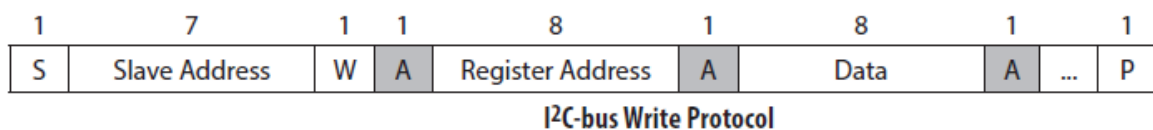
Acknowledge: Bei einer Schreib Sequenz quittiert der I²C Slave nach Erkennen seiner Slave Adresse bzw. nach jedem geschriebenen Daten Byte mit einem Acknowledge. Bei einer Lese

Sequenz quittiert der I²C Slave nach Erkennen seiner Slave Adresse mit Acknowledge. Nach jedem gelesenen Daten Byte quittiert der I²C Master mit einem Acknowledge dem Slave, dass er bereit ist, weitere Daten zu empfangen. Dabei wird zu einem ebenfalls vom Slave generierten Takt Impuls die Daten Leitung auf LOW gehalten.



No Acknowledge: Bei einer Lese Sequenz sendet der I²C Master nach dem Lesen des letzten Daten Byte ein No Acknowledge. Das bedeutet, er möchte keine weiteren Daten mehr lesen. Dabei wird zu einem ebenfalls vom Slave generierten Takt Impuls die Daten Leitung auf HIGH gehalten.

Stopp Bedingung: Die Stop Bedingung kennzeichnet das Ende einer Datenübertragung durch einen I²C Master. Der Master zieht die Datenleitung SDA von LOW auf HIGH, während die Taktleitung SCL auf HIGH bleibt.

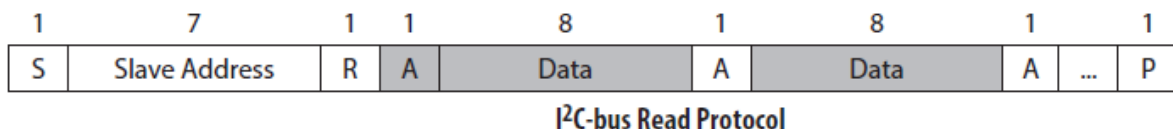
2.6.2 Write Befehl





Für den Schreibe Befehl beginnt die Anfrage mit einem Start-Bit und der Adresse des entsprechenden Slaves, welcher angesprochen werden soll und einem Write Bit (0). Wenn dies funktioniert gibt es vom Slave ein Acknowledge zurück und das Register kann angesprochen werden. Nun gibt es wieder ein Acknowledge zurück und das Register ist bereit Daten zu empfangen. Wenn dies abgeschlossen ist gibt es wieder ein Acknowledge zum Master und mit dem Stopp Bit wird die Übertragung beendet.

A	Acknowledge (0)
N	Not Acknowledged (1)
P	Stop Condition
R	Read (1)
S	Start Condition
Sr	Repeated Start Condition
W	Write (0)
...	Continuation of protocol
	Master-to-Slave
	Slave-to-Master

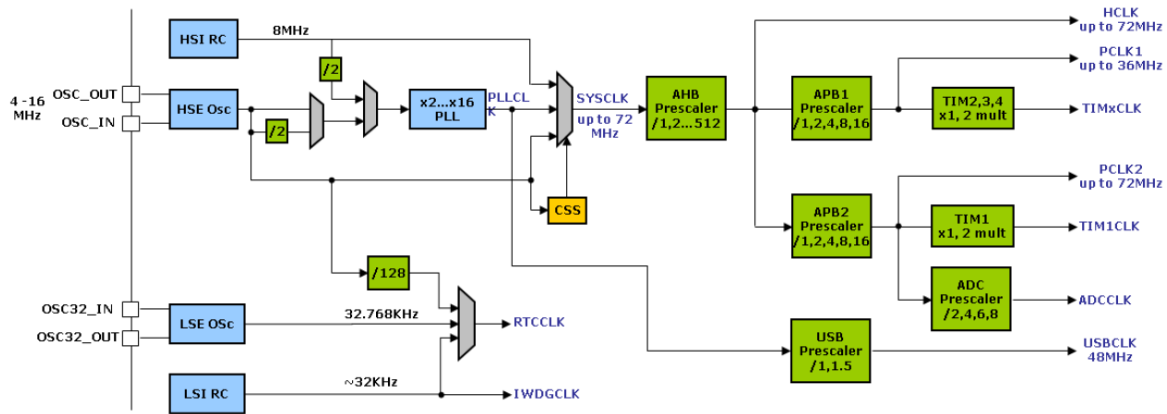
2.6.3 Read Befehl



Für den Read Befehl wird ein Start Bit gesendet mit der entsprechenden Slave Adresse und dem Read Bit (1). Hat die funktioniert, wird ein Acknowledge und die Daten zurückgesendet. Der Master bestätigt immer nach einem Byte die Daten und ein Weiteres Byte wird gesendet.

A	Acknowledge (0)
N	Not Acknowledged (1)
P	Stop Condition
R	Read (1)
S	Start Condition
Sr	Repeated Start Condition
W	Write (0)
...	Continuation of protocol
	Master-to-Slave
	Slave-to-Master

2.7 Systemtakt HSE 36MHz



Ausschnitt aus Punkt 4.2.4 aus dem DIC 4. Klasse Skriptum von Hr. Wihsböck

Um als Systemtakt den HSE mit 36MHz einzustellen mussten einige Vorkehrungen getroffen werden. Um nämlich den PLL und den HSE umstellen zu können muss zuerst der Systemtakt auf den HSI wechseln. Anschließend wird der PLL ausgeschaltet und angepasst. Dafür wird er mittels dem PLLXTPRE Register halbiert und mit dem PLLMULL Register mit Neun multipliziert und als Systemtakt definiert. Der PLL und der HSE werden wieder eingeschaltet und es wird gewartet, bis beide stabil sind.

Folgende Rechnung ergibt sich dadurch:

$$\frac{8\text{MHz (HSE)}}{2 \text{ (PLLXTPRE)}} * 9 \text{ (PLLMULL9)} = 36\text{MHz}$$

Bemerkung: Der genaue Code für den beschriebenen Vorgang findet man im Unterprogramm `void set_clock_36MHz();`

2.8 Uhr

Ein Punkt der Aufgabenstellung war es, eine Uhr mit Interrupts zu programmieren, welche beginnt zu laufen, wenn das Programm gestartet wird. Die Anzeige erfolgt in dem Format hh:mm:ss:z. Daher musste ein Interrupt alle 0,1 Sekunden ausgelöst werden, um dadurch eine Millisekunden Variable zu inkrementieren. Dafür wurde der Timer 3 verwendet. Um diesen zu konfigurieren, wurde zuerst dieser aktiviert und PSC (prescaler) und ARR (auto-reload Wert) berechnet.

Berechnung für 36MHz ($T_{\text{gesucht}}=100\text{ms}$):

$$T_{CK_INT} = \frac{1}{f_{CK_INT}} = \frac{1}{36\text{MHz}} = 27,78\text{ns}$$

$$ARR = \frac{T_{\text{gesucht}}}{T_{CK_INT}} = \frac{100\text{ms}}{27,78\text{ns}} = \sim 3\,600\,000 \Rightarrow 0xFFFF$$

$$PSC = \frac{T_{\text{gesucht}}}{T_{CK_INT} * 65535} - 1 = \frac{100\text{ms}}{27,78\text{ns} * 65535} - 1 = 53,93 \Rightarrow 54$$

Berechnung für 8MHz ($T_{\text{gesucht}}=100\text{ms}$):

$$T_{CK_INT} = \frac{1}{f_{CK_INT}} = \frac{1}{8\text{MHz}} = 125\text{ns}$$

$$ARR = \frac{T_{\text{gesucht}}}{T_{CK_INT}} = \frac{100\text{ms}}{125\text{ns}} = \sim 800\,000 \Rightarrow 0xFFFF$$

$$PSC = \frac{T_{\text{gesucht}}}{T_{CK_INT} * 65535} - 1 = \frac{100\text{ms}}{125\text{ns} * 65535} - 1 = \sim 11$$

2.9 Gestensensor

Die anfängliche Aufgabe bestand darin, von dem Sensor, der auch Gesten erkennen kann, die Daten entsprechend auszuwerten und anzugeben, welche Geste gemacht wird. Dies hat sich als zu kompliziert herausgestellt, da es noch dazu keine passende vorgefertigte Library für diesen Sensor gibt (nur für Arduino). Daher wurde anschließend die Aufgabestellung angepasst und nur der Abstandssensor verwendet.

2.10 Abstandssensor

Die Ergebnisse von dem Sensor sind durch drei Hauptfaktoren beeinflusst. Und zwar von der Infrarot LED-Abstrahlung, von dem Infrarot Licht das empfangen wird, und Umgebungseinflüsse wie die Distanz.

2.10.1 Register

Um den Sensor zu konfigurieren wurden folgende Register entsprechend gesetzt:

Table 1. Proximity Controls

Register/Bit	Address	Description
ENABLE<PON>	0x80<0>	Power ON
ENABLE<PEN>	0x80<2>	Proximity Enable
ENABLE<PIEN>	0x80<5>	Proximity Interrupt Enable
PILT	0x89	Proximity low threshold
PIHT	0x8B	Proximity high threshold
PERS<PPERS>	0x8C<7:4>	Proximity Interrupt Persistence
PPULSE<PPLEN>	0x8E<7:6>	Proximity Pulse Length
PPULSE<PPULSE>	0x8E<5:0>	Proximity Pulse Count
CONTROL<PGAIN>	0x8F<3:2>	Proximity Gain Control
CONTROL<LDRIVE>	0x8F<7:6>	LED Drive Strength
CONFIG2<PSIEN>	0x90<7>	Proximity Saturation Interrupt Enable
CONFIG2<LEDBOOST>	0x90<5:4>	Proximity/Gesture LED Boost
STATUS<PGSAT>	0x93<6>	Proximity Saturation
STATUS<PINT>	0x93<5>	Proximity Interrupt
STATUS<PVALID>	0x93<1>	Proximity Valid
PDATA	0x9C	Proximity Data
POFFSET_UR	0x9D	Proximity Offset UP/RIGHT
POFFSET_DL	0x9E	Proximity Offset DOWN/LEFT
CONFIG3<PCMP>	0x9F<5>	Proximity Gain Compensation Enable
CONFIG3<PMSK_U>	0x9F<3>	Proximity Mask UP Enable
CONFIG3<PMSK_D>	0x9F<2>	Proximity Mask DOWN Enable
CONFIG3<PMSK_L>	0x9F<1>	Proximity Mask LEFT Enable
CONFIG3<PMSK_R>	0x9F<0>	Proximity Mask RIGHT Enable
PICLEAR	0xE5	Proximity Interrupt Clear
AICLEAR	0xE7	All Non-Gesture Interrupt Clear

Nachfolgend ist eine kurze Erklärung zu jedem Register, die genaue Konfiguration ist im Programmcode nachzulesen.

2.10.2 Enable Register (0x80)

Enable Register (0x80)

The ENABLE register is used to power the device on/off, enable functions and interrupts.

Field	Bits	Description
Reserved	7	Reserved. Write as 0.
GEN	6	Gesture Enable. When asserted, the gesture state machine can be activated. Activation is subject to the states of PEN and GMODE bits.
PIEN	5	Proximity Interrupt Enable. When asserted, it permits proximity interrupts to be generated, subject to the persistence filter settings.
AIEN	4	ALS Interrupt Enable. When asserted, it permits ALS interrupts to be generated, subject to the persistence filter settings.
WEN	3	Wait Enable. This bit activates the wait feature. Writing a one activates the wait timer. Writing a zero disables the wait timer.
PEN	2	Proximity Detect Enable. This field activates the proximity detection. Writing a one activates the proximity. Writing a zero disables the proximity.
AEN	1	ALS Enable. This field activates ALS function. Writing a one activates the ALS. Writing a zero disables the ALS.
PON	0	Power ON. This field activates the internal oscillator to permit the timers and ADC channels to operate. Writing a one activates the oscillator. Writing a zero disables the oscillator and puts the device into a low power sleep mode. During reads and writes over the I2C interface, this bit is temporarily overridden and the oscillator is enabled, independent of the state of PON.

Gesetzt werden muss das Bit 5 PIEN immer dann, wenn ein Interrupt ausgelöst werden soll, also wenn ein der Sensor einen Wert innerhalb der oberen und unteren Grenze erkennt. Mit PEN wird der Sensor für den Abstand aktiviert und mit PON der ganze Sensor mit Strom versorgt.

2.10.3 Proximity Interrupt Threshold Register (0x89/0x8B)

Proximity Interrupt Threshold Register (0x89/0x8B)

The Proximity Interrupt Threshold Registers set the high and low trigger points for the comparison function which generates an interrupt. If PDATA, the value generated by proximity channel, crosses below the lower threshold specified, or above the higher threshold, an interrupt may be signaled to the host processor. Interrupt generation is subject to the value set in persistence (PERS).

Field	Address	Bits	Description
PILT	0x89	7:0	This register provides the low interrupt threshold.
PIHT	0x8B	7:0	This register provides the high interrupt threshold.

Mit diesen Registern wird der obere und untere Grenzwert zum Detektieren festgelegt.

2.10.4 Persistence Register (0x8C)

Persistence Register (0x8C)

The Interrupt Persistence Register sets a value which is compared with the accumulated amount of ALS or Proximity cycles in which results were outside threshold values. Any Proximity or ALS result that is inside threshold values resets the count.

Separate counters are provided for proximity and ALS persistence detection.

Field	Bits	Description
PPERS	7:4	Proximity Interrupt Persistence. Controls rate of proximity interrupt to the host processor.
FIELD VALUE		INTERRUPT GENERATED WHEN...
0		Every proximity cycle
1		Any proximity value outside of threshold range
2		2 consecutive proximity values out of range
3		3 consecutive proximity values out of range
...		...
15		15 consecutive proximity values out of range

Mit diesem Register wird festgelegt, nach wie vielen Messungen der Interrupt auslösen soll.

2.10.5 Proximity Pulse Count Register (0x8E)

Proximity Pulse Count Register (0x8E)

The Proximity Pulse Count Register sets Pulse Width Modified current during a Proximity Pulse. The proximity pulse count register bits set the number of pulses to be output on the LDR pin. The Proximity Length register bits set the amount of time the LDR pin is sinking current during a proximity pulse.

Field	Bits	Description
PPLEN	7:6	Proximity Pulse Length. Sets the LED-ON pulse width during a proximity LDR pulse.
		FIELD VALUE PULSE LENGTH
		0 4 μ s
		1 8 μ s (default)
		2 16 μ s
		3 32 μ s
PPULSE	5:0	Proximity Pulse Count. Specifies the number of proximity pulses to be generated on LDR. Number of pulses is set by PPULSE value plus 1.
		FIELD VALUE NUMBER OF PULSES
		0 1
		1 2
		2 3
	
		63 64

In diesem Register wird festgelegt, wie lange der Puls für den Abstandsensor sein soll und wie viele Pulse gezählt werden müssen.

2.10.6 Control Register One (0x8F)

Control Register One (0x8F)

Field	Bits	Description
LDRIVE	7:6	LED Drive Strength.
		FIELD VALUE LED CURRENT
		0 100 mA
		1 50 mA
		2 25 mA
		3 12.5 mA
Reserved	5	Reserved. Write as 0.
Reserved	4	Reserved. Write as 0.
PGAIN	3:2	Proximity Gain Control.
		FIELD VALUE GAIN VALUE
		0 1x
		1 2x
		2 4x
		3 8x

Bei diesem Register wird die Stärke der LED und die Verstärkung eingestellt, um die Intensität mit welcher der Sensor reagiert zu steuern.

2.10.7 Configuration Register Two (0x90)

Configuration Register Two (0x90)

The Configuration Register Two independently enables or disables the saturation interrupts for Proximity and Clear channel. Saturation Interrupts are cleared by accessing the Clear Interrupt registers at 0xE5, 0xE6 and 0xE7. The LED_BOOST bits allow the LDR pin to sink more current above the maximum setting by LDRIVE and GLDRIVE.

Field	Bits	Description
PSIEN	7	Proximity Saturation Interrupt Enable. 0 = Proximity saturation interrupt disabled 1 = Proximity saturation interrupt enabled
CPSIEN	6	Clear Photodiode Saturation Interrupt Enable. 0 = ALS Saturation Interrupt disabled 1 = ALS Saturation Interrupt enabled
LED_BOOST	5:4	Additional LDR current during proximity and gesture LED pulses. Current value, set by LDRIVE, is increased by the percentage of LED_BOOST.
		FIELD VALUE LED BOOST CURRENT
		0 100%
		1 150%
		2 200%
		3 300%

Mit diesem Register kann der Strom durch die LED nochmals verstärkt werden, um Gesten, die weiter weg sind, zu erkennen.

2.10.8 Status Register (0x93)

Status Register (0x93)

The read-only Status Register provides the status of the device. The register is set to 0x04 at power-up.

Field	Bits	Description
CPSAT	7	Clear Photodiode Saturation. When asserted, the analog sensor was at the upper end of its dynamic range. The bit can be de-asserted by sending a Clear channel interrupt command (0xE6 CICLEAR) or by disabling the ADC (AEN=0). This bit triggers an interrupt if CPSIEN is set.
PGSAT	6	Indicates that an analog saturation event occurred during a previous proximity or gesture cycle. Once set, this bit remains set until cleared by clear proximity interrupt special function command (0xE5 PICLEAR) or by disabling Prox (PEN=0). This bit triggers an interrupt if PSIEN is set.
PINT	5	Proximity Interrupt. This bit triggers an interrupt if PIEN in ENABLE is set.
AINT	4	ALS Interrupt. This bit triggers an interrupt if AIEN in ENABLE is set.
RESERVED	3	Do not care.
GINT	2	Gesture Interrupt. GINT is asserted when GFVLV becomes greater than GFIFOTH or if GVALID has become asserted when GMODE transitioned to zero. The bit is reset when FIFO is completely emptied (read).
PVALID	1	Proximity Valid. Indicates that a proximity cycle has completed since PEN was asserted or since PDATA was last read. A read of PDATA automatically clears PVALID.
AVALID	0	ALS Valid. Indicates that an ALS cycle has completed since AEN was asserted or since a read from any of the ALS/Color data registers.

Das Status Register kann nur ausgelesen werden und stellt die Informationen zum Status des Geräts bereit.

2.10.9 Proximity Data Register (0x9C)

Proximity Data Register (0x9C)

Proximity data is stored as an 8-bit value.

Field	Address	Bits	Description
PDATA	0x9C	7:0	Proximity data.

In diesem Register werden die Daten von dem Abstandssensor gespeichert.

2.10.10 Configuration Three Register (0x9F)

Configuration Three Register (0x9F)

The CONFIG3 register is used to select which photodiodes are used for proximity. Two photodiodes are paired to provide signal. In proximity mode, UP and RIGHT photodiodes are connected forming a diode pair; similarly the DOWN and LEFT photodiodes form a diode pair.

Field	Bits	Description
RESERVED	7:6	Reserved. Write as 0.
PCMP	5	Proximity Gain Compensation Enable. This bit provides gain compensation when proximity photodiode signal is reduced as a result of sensor masking. If only one diode of the diode pair is contributing, then only half of the signal is available at the ADC; this results in a maximum ADC value of 127. Enabling PCMP enables an additional gain of 2X, resulting in a maximum ADC value of 255.
PMASK_X (U, D, L, R)		PCMP
0, 1, 1, 1		1
1, 0, 1, 1		1
1, 1, 0, 1		1
1, 1, 1, 0		1
0, 1, 0, 1		1
1, 0, 1, 0		1
All Others		0
SAI	4	Sleep After Interrupt. When enabled, the device will automatically enter low power mode when the INT pin is asserted and the state machine has progressed to the SAI decision block. Normal operation is resumed when INT pin is cleared over I2C.
PMASK_U	3	Proximity Mask UP Enable. Writing a 1 disables this photodiode.
PMASK_D	2	Proximity Mask LEFT Enable. Writing a 1 disables this photodiode.
PMASK_L	1	Proximity Mask LEFT Enable. Writing a 1 disables this photodiode.
PMASK_R	0	Proximity Mask RIGHT Enable. Writing a 1 disables this photodiode.

Bei diesem Register könnte man bestimmte Fotodioden deaktivieren.

2.11 Testprogramm

Das Programm (main.c, proximity.c, proximity.h => Source Code siehe Anhang) liest über den Sensor die Distanz zu einem Objekt aus und gibt diese in einer nicht linearen Einheit an (siehe). Die ersten zwei Sekunden wird jedoch ein Begrüßungstext („HELLO“) ausgegeben und es wird überprüft, ob der Sensor mit dem Mikrocontroller verbunden ist oder nicht.

Die Ausgabe am LCD sieht folgenden Maßen aus:

```
HELLO  
DEVICE DIS/CONNECTED
```

Zusätzlich wird im Programm eine Range festgelegt (z.B. 16 - 172) und es wird angezeigt, ob das gemessene Objekt innerhalb dieser Range liegt. Außerdem startet nach dem Begrüßungstext ein Timer, welcher angibt, wie lange das Programm schon läuft. Dieser wird nach jedem Reset neu gestartet. Der Text am LCD sieht folgender Maßen aus:



Hierbei liegt das gemessene Objekt (in diesem Fall einen Ordner vor dem Sensor) in einer Entfernung von 91 Punkten und liegt somit innerhalb der festgelegten Range.

2.11.1 Konfiguration

Wie bereits erwähnt müssen am Anfang des Programms die verschiedenen Register wie folgt konfiguriert werden:

Enable Register (Regaddr = 0x80): 00000101b (Data = 0x05)

- Bit 7 = 0: is being reserved as 0
- Bit 6 = 0: gesture enable (GEN)
- Bit 5 = 0: proximity interrupt enable (PIEN)
- Bit 4 = 0: ambient light sense (ALS) interrupt enable (AIEN)
- Bit 3 = 0: wait enable (WEN) activates wait feature
- Bit 2 = 1: proximity detect enable (PEN)
- Bit 1 = 0: ALS enable (AEN)
- Bit 0 = 1: Power ON (PON)

Persistence Register (Regaddr = 0x8C): 00000000b (Data = 0x0)

- Bit 7 : 4 = 0: Controls rate of proximity interrupt to host process
- Bit 3 : 0 = 0: Controls rate of clear channel interrupt to host process

Proximity Interrupt Threshold Register Low (Regaddr = 0x89): 00010000b (Data = 0x10):

- Bit 7 : 0 = adjustable

Proximity Interrupt Threshold Register High (Regaddr = 0x8B): 10101100b (Data = 0xAC)

Bit 7 : 0 = adjustable

Proximity Pulse Register (Regaddr = 0x8E): 01111111b (Data = 0x7F)

Bit 7 : 6 = 01

Bit 5 : 0 = 1: 111111

Control Register One (Regaddr = 0x8F): 00001000b (Data = 0x08)

Bit 7 : 6 = 10

Bit 5 : 4 = reserved as 0

Bit 3 : 2 = adjustable

Bit 1 : 0 = 0

Control Register Two (Regaddr = 0x90): 10100000b (Data = 0x80)

Bit 7 = 1

Bit 6 = 0

Bit 5 : 4 = adjustable

Bit 3 : 0 = reserved as 0

Status Register (Regaddr = 0x93): 01100010b (Data = 0x62)

Bit 7 = 0 Clear Photodiode Saturation (CPSAT)

Bit 6 = 1 Indicates that an analog saturation event occurred (PGSAT)

Bit 5 = 1 Proximity Interrupt. This bit triggers an interrupt if PIEN in ENABLE is set (PINT)

Bit 4 = 0 ALS Interrupt (AINT)

Bit 3 = 0 reserved as 0

Bit 2 = 0 Gesture Interrupt (GINT)

Bit 1 = 1 Proximity Valid (PVALID)

Bit 0 = 0 ALS Valid (AVALID)

Proximity Offset UP / RIGHT Register (Regaddr = 0x9D): 00000000b (Data = 0x00)

Bit 7 : 0 = 0x0

Proximity Offset DOWN / LEFT Register (Regaddr = 0x9E): 00000000b (Data = 0x00)

Bit 7 : 0 = 0x0

Configuration Register Three (Regaddr = 0x9F): 00000000b (Data = 0x00)

Bit 7 : 6 = reserved as 0

Bit 5 = 0 use all diodes

Bit 4 = 0 sleep after interrupt

Bit 3 = 0 Proximity Mask UP Enable

Bit 2 = 0 Proximity Mask LEFT Enable

Bit 1 = 0 Proximity Mask LEFT Enable

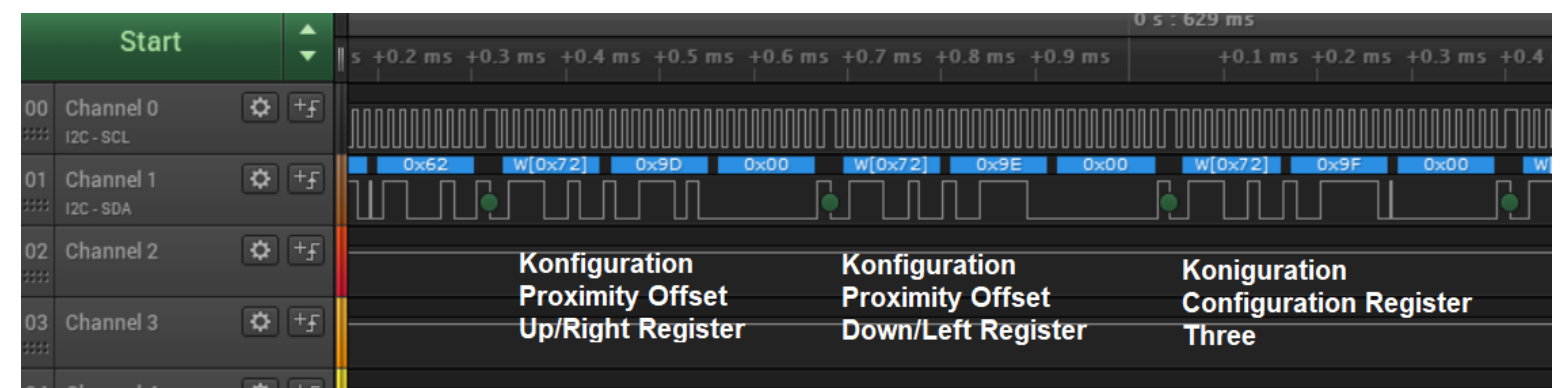
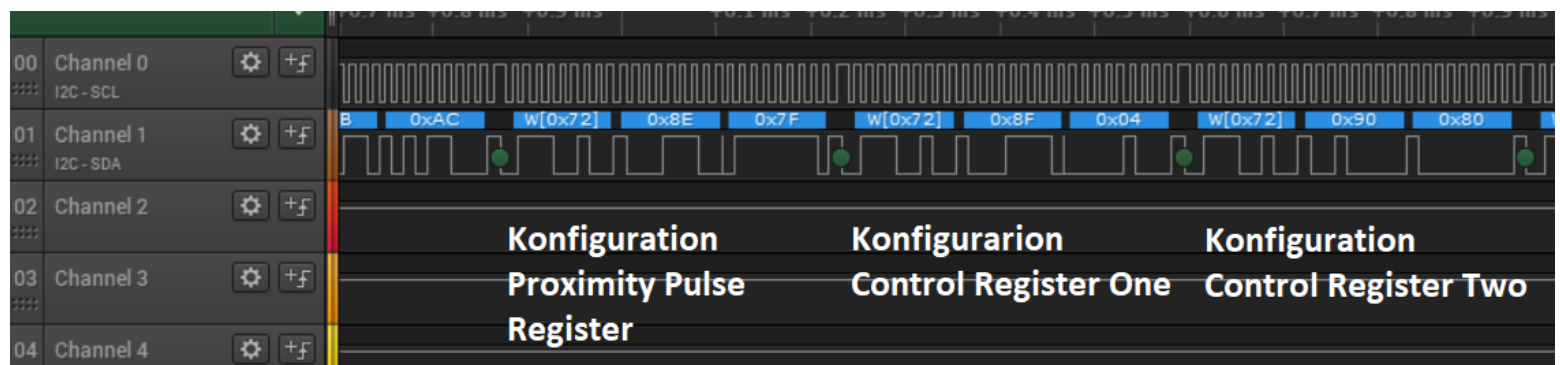
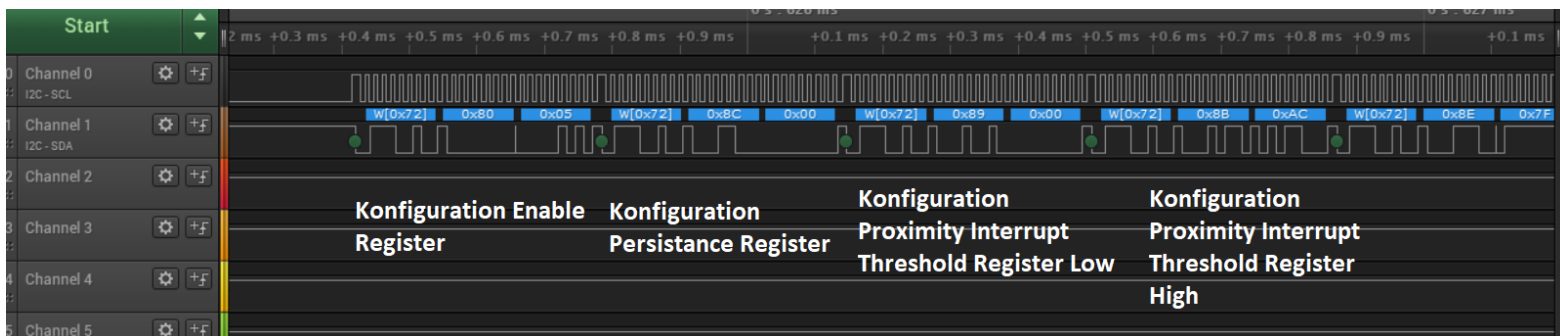
Bit 0 = 0 Proximity Mask RIGHT Enable

Proximity Interrupt Clear (Regaddr = 0xE5): 00000000b (Data = 0x00)

Bit 7 : 0 = 0x0

Clear All Non-Gesture Interrupts (Regaddr = 0xE7): 00000000b (Data = 0x00)

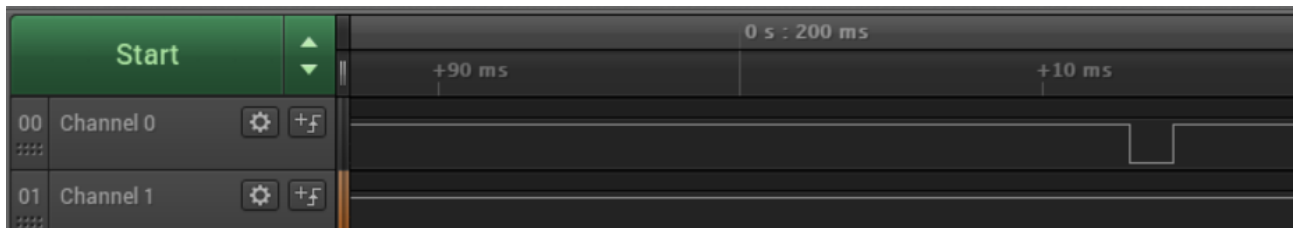
Bit 7 : 0 = 0x0



Auf diesen Bildern sieht man die Konfiguration der verschiedenen Register im Logik Analysator.

2.11.2 Externer Interrupt PA1

Der INT Pin des Sensors ist mit unserer Portleitung PA1 direkt verbunden, welcher als externer Interrupt Eingang konfiguriert und festgelegt ist. Jedes Mal, wenn das gemessene Objekt innerhalb der Range liegt, führt der Sensor ein Interrupt aus, indem das PIEN bit des ENABLE REGISTERs gesetzt wird. Unser Programm erkennt diesen Interrupt als steigende Flanke und gibt den oben genannten Text am LCD aus.



Auf diesem Bild sieht man den Pin PA1 auf dem Logikanalysator, während der Sensor einen Impuls auslöst, da ein Objekt in die vorgegebene Range gelangt ist.

2.12 Probleme

- Fehler beim Setzen der Interrupts
- Fehler beim Konfigurieren der 36MHz, da man HSE und PLL ausschalten muss
- Fehler mit der Library von Jakob Pachtrog, beim Einstellen des Systemtakts auf 36MHz
- Probleme beim Verstehen des Datenblattes
- Probleme bei schöner Ausgabe auf dem LC Display

2.13 Erkenntnisse

- Verwendung eines Logikanalysators
- Verwendung eines Bussystems und das bitweise Senden in bestimmten zeitlichen Abständen
- Passende Informationsfindung über Datenblätter und dergleichen
- Verwenden einer vorgefertigten Library
- Funktionsweise des I²C-Bus Protokolls
- Funktionsweise und Verwendung von Interrupts
- Funktionsweise des Timers und Berechnungen von PSC und ARR

2.14 Zeitaufwand – Stefan Grubmüller

Tätigkeit	Aufwand
Erstellung des Pflichtenhefts	1h
Programmcodierung	26h
Testen der Software	4h
Dokumentation (Protokoll)	2h
Gesamt:	33h

2.15 Zeitaufwand – Clemens Marx

Tätigkeit	Aufwand
Erstellung des Pflichtenhefts	1h
Programmcodierung	26h
Testen der Software	2h
Dokumentation (Protokoll)	4h
Gesamt:	33h