IMS
Institut für Mikroelektronische Systeme
Leibniz Universität Hannover

Leibniz Universität Hannover

Seminararbeit

Implementierung eines Echtzeit-Demonstrators für eine Computed Order Tracking Anwendung

Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Mikroelektronische Systeme
Fachgebiet Architekturen und Systeme

vorgelegt von

Henning Piper, B.Sc.
Matrikelnummer 10001838
geb. am: 26.01.1995 in: Lübbecke

Betreuer: M.Sc. Jens Karrenbauer

Hannover, 4. Februar 2022

IMS
Institut für Mikroelektronische Systeme
Leibniz Universität Hannover

Leibniz Universität Hannover

Hannover, 30.11.2021

SEMINARARBEIT

für Herrn Henning Piper

Implementierung eines Echtszeit-Demonstrators für eine Computed Order Tracking Anwendung

Durch den digitalen Fortschritt und Industrie 4.0 werden Produktionsmaschinen immer größer und komplexer. Um die neu hinzugekommenen Systemfunktionen zu überwachen sind die Geräte stärker vernetzt und mit einer höheren Anzahl von Sensoren ausgestattet. Aufgrund der beweglichen und metallischen Elemente in solchen Maschinen können weder große Kabelbäume noch drahtlose Verbindungen die Sensordaten transportieren. Daher ist eine Vorverarbeitung der Daten und eine Sensorfusion direkt an der Quelle erforderlich. Da jedoch der Platz und der Energieverbrauch solcher Maschinen begrenzt sind, muss die Architektur für die Signalverarbeitung so klein und energieeffizient wie möglich sein. Weiterhin müssen diese Systeme eine Echtzeitfähigkeit gewährleisten, um rechtzeitig z. B. auf sicherheitsrelevante Störungen reagieren zu können.

Im Fachgebiet "Architekturen und Systeme" des Instituts für Mikroelektronische Systeme werden dedizierte und programmierbare Architekturen mit speziellen Anforderungen an Echtzeitfähigkeit und Energieverbrauch entworfen, implementiert und getestet. Eine geeignete Architektur, die diese Anforderungen erfüllt, ist ein anwendungsspezifischer Befehlssatzprozessor (ASIP). Diese Architektur kann für bestimmte Anwendungsklassen optimiert werden, indem spezielle Hardwareeinheiten erweitert werden, um die Rechenleistung zu erhöhen oder den Energieverbrauch zu senken. Um diese Ziele zu erreichen, wird nicht nur die Hardware analysiert, es werden auch die Signalverarbeitungsalgorithmen der Anwendungsklassen evaluiert und für die Zielarchitektur optimiert. Durch die Kombination dieser Methoden kann ein echtzeitfähiges, platzsparendes und energieeffizientes komplexes Filtersystem für die Sensordatenverarbeitung erforscht werden.

Im Rahmen dieser Arbeit soll Herr Piper eine Referenzimplementierung einer Computed Order Tracking Anwendung für einen Cadence Tensilica Fusion G3 analysieren und diese auf den Cadence Tensilica Fusion G6 portieren. Anschließend soll die Anwendung auf dem SmartHeaP SoC in Kombination mit dem dazugehörigen Bringup Bord emuliert werden. Dafür muss im ersten Schritt die Anwendung echtzeitfähig sein. Weiterhin sollen die verwendeten Simulationsinterfaces der Referenz durch serielle Interfaces des SoCs ausgetauscht werden. Zur Simulation der Sensorausgangsdaten soll der bereitgestellte Arduino MKR ZERO verwendet werden.

Abschließend ist auf gute Dokumentation zu achten. Das eingereichte Exemplar sowie die Ergebnisse der Arbeit bleiben Eigentum des Instituts.

Prof. Dr.-Ing. Holger Blume

Prof. Dr.-Ing Holger Blume
blume@ims.uni-hannover.de

Appelstraße 4 | 30167 Hannover
fon 0511 762-19640 | fax 0511 762-19601
www.ims.uni-hannover.de | info@ims.uni-hannover.de

Erklärung

Ich versichere hiermit, dass ich die vorstehende Arbeit selbständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und sowohl wörtliche, als auch sinngemäßentlehnte Stellen als solche kenntlich gemacht habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

H. Piper

Hannover, den 4. Februar 2022

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Sensor KX134-1211

Für die Erfassung der Beschleunigungsdaten in X-, Y- und Z-Richtung wird der Beschleunigungssensor KX134-1211 verwendet. In den folgenden Abschnitten sind die wichtigsten Sensordaten, die bei der Einbindung des Sensors in den Demonstrator verwendet wurden, zu finden. Hierzu zählen die Pinbelegung sowie Pinbeschreibung in Abschnitt 1.1, die Registermap aller 128 Sensorregister in Abschnitt 1.2, sowie der Ablauf eines Lese- bzw. Schreibzykluses des Sensors in Abschnitt 1.3. Abschließend wird in Abschnitt 1.4 auf ein paar mögliche Initialisierungsoptionen des Sensors für beispielsweise das synchrone oder asynchrone Auslesen von Daten eingegangen. Alle hier aufgeführten Daten sind [1], [2], [3] und [4] entnommen.

1.1 Pinbelegung und -beschreibung

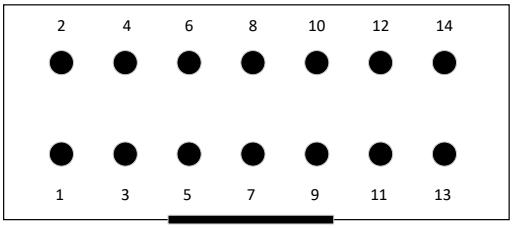

Abbildung 1.1: Pinbelegung des Eval-Kits des KX134-1211

Tabelle 1.1: Pinbelegung des Eval-Kits des KX134-1211, [1, p.10]

Pin	Bezeichnung	Beschreibung
1	VDD	Betriebsspannung
2	nCs	Chip Select (active Low) für SPI Kommunikation, Mit IO_VDD verbinden falls eine I²C Kommunikation genutzt wird ! Pin darf nicht floaten !
3	N.C.	Not Connected, Kann mit VDD, IO_VDD oder GND verbunden werden, alternativ darf dieser Pin auch floaten
4	N.C.	Not Connected, Kann mit VDD, IO_VDD oder GND verbunden werden, alternativ darf dieser Pin auch floaten
5	SCLK/SCL	SPI und I²C serial clock
6	IO_VDD	Spannungsversorgung für den digitalen Kommunikationsbus, wird mit VDD verbunden
7	SDI/SDA	SPI Dateneingangspin, I²C Datenpin
8	GND	Masse
9	SDO/ADDR	Serial Data Out Pin während einer SPI-Kommunikation, außerdem kann über diesen Pin die Sensoradresse für die I²C Kommunkation beeinflusst werden

10	TRIG	Trigger Pin für die FIFO Buffer Kontrolle, Mit GND verbinden falls die externe Trigger Option nicht genutzt wird
11	INT1	Physical Interrupt 1 (Push-Pull) Dieser Pin befindet sich im High-Z State während des POR und wechselt in einen niederohmigen Zustand nach dem POR (Power on reset)
12	INT2	Physical Interrupt 2 (Push-Pull) Dieser Pin befindet sich im High-Z State während des POR und wechselt in einen niederohmigen Zustand nach dem POR (Power on reset)
13	N.C.	Not Connected, Kann mit VDD, IO_VDD oder GND verbunden werden, alternativ darf dieser Pin auch floaten
14	N.C.	Not Connected, Kann mit VDD, IO_VDD oder GND verbunden werden, alternativ darf dieser Pin auch floaten

1.2 Registermap

Der Beschleunigungssensor enthält 128 Register à 8-Bit, welche überwiegend frei programmierbar durch den Nutzer sind. Die folgende Tabelle gibt einen kurzen Überblick über alle Register wobei die für diese Anwendung genutzten Register farblich gekennzeichnet sind.

Tabelle 1.2: Registermap des KX134-1211,[2, p.6]

Adr	Register	R/W	Adr	Register	R/W	Adr	Register	R/W
00	MAN_ID	R	1B	CNTL1	R/W	33	FFC	R/W
01	PART_ID	R	1C	CNTL2	R/W	34	FFCNTL	R/W
02	XADP_L	R	1D	CNTL3	R/W	35-36	Reserviert	-
03	XADP_H	R	1E	CNTL4	R/W	37	TILT_ANGLE_LL	R/W
04	YADP_L	R	1F	CNTL5	R/W	38	TILT_ANGLE_HL	R/W
05	YADP_H	R	20	CNTL6	R/W	39	HYST_SET	R/W
06	ZADP_L	R	21	ODCNTL	R/W	3A	LP_CNTL1	R/W
07	ZADP_H	R	22	INC1	R/W	3B	LP_CNTL2	R/W
08	XOUT_L	R	23	INC2	R/W	3C-48	Reserviert	-
09	XOUT_H	R	24	INC3	R/W	49	WUFTH	R/W
0A	YOUT_L	R	25	INC4	R/W	4A	BTSWUFTH	R/W
0B	YOUT_H	R	26	INC5	R/W	4B	BTSTH	R/W
0C	ZOUT_L	R	27	INC6	R/W	4C	BTSC	R/W
0D	ZOUT_H	R	28	Reserviert	-	4D	WUFC	R/W
0E-11	Reserviert	-	29	TILT_TIMER	R/W	4E-5C	Reserviert	-
12	COTR	R	2A	TDTRC	R/W	5D	SELF_TEST	W
13	WHO_AM_I	R	2B	TDTC	R/W	5E	BUF_CNTL1	R/W
14	TSCP	R	2C	TTH	R/W	5F	BUF_CNTL2	R/W
15	TSPP	R	2D	TTL	R/W	60	BUF_STATUS_1	R
16	INS1	R	2E	FTD	R/W	61	BUF_STATUS_2	R
17	INS2	R	2F	STD	R/W	62	BUF_CLEAR	W
18	INS3	R	30	TLT	R/W	63	BUF_READ	R
19	STATUS_REG	R	31	TWS	R/W	64-76	ADP_CNTL (1-19)	R/W
1A	INT_REL	R	32	FFTH	R/W	77-7F	Reserviert	-

Beschleunigungsdatenregister (0x08-0x0D):

Sobald der Sensor aktiviert ist, werden für die 16-Bit Daten der drei Beschleunigungsachsen in den Registern XOUT_L bis ZOUT_H abgespeichert. Die Daten werden dabei periodisch aktualisiert, die entsprechende Periode kann vom Nutzer über die Bits 3:0 im Register ODCNTL gesetzt werden. Je nachdem ob von den Beschleunigungsdaten die vollständigen 16 Bit oder nur die höchst oder niederwertigsten 8-Bit ausgelesen werden, ändert sich dementsprechend auch die damit verbundene Auflösung der Daten [2, p.8 ff.]. Die Daten liegen im Zweierkomplement vor und können für die weitere Nutzung in Beschleunigungswerte umgewandelt werden, entsprechend der folgenden Tabellen 1.3 und 1.4.

Tabelle 1.3: Wertebereich der Beschleunigungsdaten bei einer 16-Bit Auflösung, [2, p.10]

16-Bit Register Daten	Äquivalenter Dezimalbetrag	Auflösung ±8 g	Auflösung ±16 g	Auflösung ±32 g	Auflösung ±64 g
0111 1111 1111 1111 1111	32767	+7.99976g	+15.99951g	+31.99902g	+63.99805g
0111 1111 1111 1111 1110	32766	+7.99951g	+15.99902g	+31.99805g	+63.99609g
...	...	...	...	...	...
0000 0000 0000 0000 0001	1	+0.00024g	+0.00049g	+0.00098g	+00195g
0000 0000 0000 0000 0000	0	0.00000g	0.00000g	0.00000g	0.00000g
1111 1111 1111 1111 1111	-1	-0.00024g	-0.00049g	-0.00098g	-0.00195g
...	...	...	...	...	...
1000 0000 0000 0000 0001	-32767	-7.99976g	-15.99951g	-31.99902g	-63.99805g
1000 0000 0000 0000 0000	-32768	-8.00000g	-16.00000g	-32.00000g	-64.00000g

Tabelle 1.4: Wertebereich der Beschleunigungsdaten bei einer 8-Bit Auflösung, [2, p.10]

8-Bit Register Daten	Äquivalenter Dezimalbetrag	Auflösung ±8 g	Auflösung ±16 g	Auflösung ±32 g	Auflösung ±64 g
0111 1111	127	+7.93750g	+15.87500g	+31.75000g	+63.50000g
0111 1110	126	+7.87500g	+15.75000g	+31.50000g	+63.00000g
...	...	...	...	...	...
0000 0001	1	+0.06250g	+0.12500g	+0.25000g	+0.50000g
0000 0000	0	0.0000g	0.0000g	0.0000g	0.0000g
1111 1111	-1	-0.06250g	-0.12500g	-0.25000g	-0.50000g
...	...	...	...	...	...
1000 0001	-127	-7.93750g	-15.87500g	-31.75000g	-63.50000g
1000 0000	-128	-8.00000g	-16.00000g	-32.00000g	-64.00000g

Kontrollregister (0x1B-0X20):

Über diese Register können einige der Hauptfunktionen initialisiert und nutzerspezifisch eingestellt werden. Für diese Anwendung zunächst am wichtigsten ist dabei das Register CNTL1, da dieses die Hauptfunktionen beeinflusst [2, p.18].

! Wichtig hierbei ist es, das Kontrollbit PC1 dieser Register zunächst auf „0“ zu setzen, bevor die Inhalte der Register umgeschrieben werden. Andernfalls kann es zu unvorhersehbarem Verhalten des Sensors kommen!

Tabelle 1.5: Bitbelegung des Registers CNTL1

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
PC1	RES	DRDYE	GSEL1	GSEL0	TDTE	Reserviert	TPE	Wert nach Reset
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	00000000

- PC1 : Schaltet den Sensor ein oder versetzt ihn in den Standby Modus.

  0 = Standby Mode
  1 = Sensor aktiv

- RES : Bestimmt den Performance Modus des Sensors. Das Rauschen der Daten wird beeinflusst durch ODR, RES und verschiedenen Einstellungen des Registers LP_CNTL1. Dies resultiert in einer höheren oder niedrigeren Auflösung des Sensors.

  0 = Low Power Mode (höheres Rauschen, niedrigere Stromaufnahme)
  1 = High-Performance Mode (niedrigeres Rauschen, höhere Stromaufnahme)

- DRDYE : Das "Data Ready interrupt Bit" zeigt an ob neue Beschleunigungsdaten in den Ausgaberegistern 0x08-0x0D vorliegen. Dieses Bit wird für das synchrone Auslesen von Daten genutzt und zurückgesetzt sobald Daten ausgelesen wurden oder ein Interrupt vorliegt.

  0 = Neue Beschleunigungsdaten liegen nicht vor
  1 = Neue Beschleunigungsdaten liegen vor

- GSEL <1:0> : Auflösung des Beschleunigungssensors entsprechend Tabelle 1.6.

Tabelle 1.6: Bitbelegung der GSEL Register

GSEL 1	GSEL0	Auflösung
0	0	±8g
0	1	±16g
1	0	±32g
1	1	±64g

- TDTE : Das "Tap/Double Tap Engine"Bit aktiviert oder deaktiviert die direktionale "Tap"Detektierung. Genaueres hierzu kann in [Reference Manual P63] nachgelesen werden.

  0 = Aktiviert
  1 = Deaktiviert

- Reserviert : Der Wert dieses Registers darf nicht verändert werden

- TPE : Das "Tilt Position Engine"Bit aktiviert oder deaktiviert die Neigungserkennung des Sensors.

  0 = Aktiviert
  1 = Deaktiviert

Kontrollregister ODCNTL(0x21):

Dieses Register ist verantwortlich für die Einstellungen welche die Beschleunigungsdatenregister betreffen [2, p.25].

Tabelle 1.7: Bitbelegung des Registers ODCNTL

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reserviert	LPRO	FSTUP	Reserviert	OSA3	OSA2	OSA1	OSA0	Wert nach Reset
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	00000110

- LPRO : Tiefpassfilter Roll-Off Kontrollbit

  0 = IIR Filter Grenzfrequenz wird auf ODR/9 gesetzt (Standard)
  1 = IIR Filter Grenzfrequenz wird auf ODR/2 gesetzt

- FSTUP : Fast Start Up Enable Bit

  Das setzen dieses Bits stellt die Startzeit des Sensors ein, wenn dieser im High-Performance Modus arbeitet mit einer Aktualisierungsrate von ODR <= 200 Hz. Ist diese Option deaktiviert, ist die Startzeit des Sensors im High-Performance Modus variabel und abhängig von ODR. Wird diese Option gesetzt, ist die Startzeit fix und unabhängig von ODR.

  0 = Aktiviert
  1 = Deaktiviert

- Reserviert : Der Wert dieses Registers darf nicht verändert werden

- OSA <3:0> : Über diese Bits wird die Aktualisierungsrate der Daten festgelegt. Der Standardwert beträgt 50 Hz. Der blau markierte Bereich ist dabei nur bei der Verwendung des High-Performance Modus nutzbar.

Tabelle 1.8: Bitbelegung der OSA Register

OSA3	OSA2	OSA1	OSA0	Aktualisierungsrate (Hz)
0	0	0	0	0.781
0	0	0	1	1.563
0	0	1	0	3.125
0	0	1	1	6.25
0	1	0	0	12.5
0	1	0	1	25
0	1	1	0	50
0	1	1	1	100
1	0	0	0	200
1	0	0	1	400
1	0	1	0	800
1	0	1	1	1600
1	1	0	0	3200
1	1	0	1	6400
1	1	1	0	12800
1	1	1	1	25600

1.3 Schreib- und Lesezyklus

Vor der Nutzung des Sensors muss dieser zunächst initialisiert werden. Hierzu werden die Kontrollregister des Sensors, entsprechend der gewünschten Funktionen, beschrieben. Während der Schreib- und späteren Lesevorgänge muss dabei folgendes Kommunikationsprotokoll beobachtet werden [4, p.21,22].

Legende:

S = Start Condition

Sr = Repeated Start Condition

SAD = Slave Adress

RA = Read Adress

W = Write Bit enable

R = Read Bit enable

ACK = Acknowledged

NACK = Not Acknowledged

M-Code= Master-Code

1.3.1 Schreibzyklus eines 8-Bit Registers

Fall 1: Der Master schreibt ein Byte in ein Register des Slaves.

Master	S	SAD+W		RA		DATA		P
Slave			ACK		ACK		ACK	

Fall 2: Der Master schreibt mehrere Byte in Register des Slaves

Master	S	SAD+W		RA		DATA		DATA		P
Slave			ACK		ACK		ACK		ACK	

1.3.2 Lesezyklus eines 8-Bit Registers

Fall 1: Der Master liest ein Byte aus einem Register des Slaves aus.

Master	S	SAD+W		RA		Sr	SAD+R			NACK	P
Slave			ACK		ACK			ACK	DATA		

Fall 2: Der Master liest mehrere Byte aus Registern des Slaves aus.

Master	S	SAD+W		RA		Sr	SAD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

1.3.3 Schreib-/Lesezyklus im Highspeed Modus

Um den „High Speed Mode“ (3.4 MHz) zu nutzen, muss der Sensor zunächst eine spezifische Startsequenz erhalten. Nach einer Start-Bedingung folgt ein Mastercode (00001XXX) und ein NACK des Masters. Sobald diese Nachrichtensequenz von dem Sensor erkannt wird, wechselt dieser in die HS-Kommunikation. Der Sensor wechselt wieder in den Fast-Mode nachdem eine STOP Bedingung über den Bus gesendet wird.

Fall 1: HS-Mode Datentransfer, Der Master schreibt mehrere Byte in Register des Slaves.

Speed	FS-Mode			HS-Mode								FS-Mode
Master	S	M-Code	NACK	Sr	SAD+W		RA		DATA		P	
Slave						ACK		ACK		ACK		

Fall 2: HS-Mode Datentransfer, Der Master liest mehrere Byte aus Registern des Slaves aus.

Speed	FS-Mode			HS-Mode				
Master	S	M-Code	NACK	Sr	SAD+W		RA	
Slave						ACK		ACK

Speed	HS-Mode								FS-Mode
Master	Sr	SAD+R					NACK	P	
Slave			ACK	DATA	ACK	DATA			

Hier sind (n-1) Bytes + ACK möglich

1.4 Quick-Start Guide

Je nach Anwendungsfall kann der Sensor spezifisch initialisiert und genutzt werden. Hierbei ergeben sich drei wichtige verschieden Initialisierungsmöglichkeiten (weitere Optionen können im „Getting Started Guide“ nachgeschlagen werden, [3, p.2 ff.])

1. Asynchrones Auslesen der Beschleunigungsdaten
2. Synchrones Auslesen der Beschleunigungsdaten, sobald das nächste Datentripple verfügbar ist (mit oder ohne Hardware Interrupt möglich)
3. Auslesen der Daten aus dem Sample Buffer, sobald dieser voll ist

Die entsprechenden Startsequenzen werden im folgenden kurz aufgelistet.

1.4.1 Asynchrones Auslesen

Schreibe 0x00 in das Register CNTL1 um den Sensor in den Standby-Modus zu versetzen.

Register Name	Adresse	Wert
CNTL1	0x1B	0x00

Schreibe 0x06 in das Register ODCNTL (Output Data Control) um die Ausgaberate ODR der Ausgangsdaten auf 50 Hz zu setzen (Dieser Schritt ist optional da dies den Standardeinstellungen entspricht).

Register Name	Adresse	Wert
ODCNTL	0x21	0x06

Schreibe 0xC0 in das Register CNTL1 um den Sensor einzuschalten (PC 1 = 1). Die weiteren eingestellten Optionen sind dabei: Power Mode (RES = 1), Data Ready deaktiviert (DRDYE = 0), Auflösung = ±8g (GSEL = 0).

Register Name	Adresse	Wert
CNTL1	0x1B	0xC0

Die Sensordaten können nun asynchron ausgelesen werden. Um das mehrfache auslesen der gleichen Daten zu reduzieren, sollte mindestens 1 / ODR Perioden gewartet werden bevor die nächsten Daten ausgelesen werden.

1.4.2 Synchrones Auslesen mit Hardware Interrupt

Schreibe 0x00 in das Register CNTL1 um den Sensor in den Standby-Modus zu versetzen.

Register Name	Adresse	Wert
CNTL1	0x1B	0x00

Schreibe 0x30 in das Register INC1 (Interrupt Control 1) um den Pin INT1 (Physical Interrupt) zu aktivieren, der Pin ist ein „high active“ Eingang.

Register Name	Adresse	Wert
INC1	0x22	0x30

Schreibe 0x10 in das Register INC4 (Interrupt Control 4) um den „Data Ready“ Interrupt auf dem Pin INT1 auszugeben.

Register Name	Adresse	Wert
INC4	0x25	0x10

Schreibe 0x06 in das Register ODCNTL (Output Data Control) um die Ausgaberate ODR der Ausgangsdaten auf 50 Hz zu setzen (Dieser Schritt ist optional da dies den Standardeinstellungen entspricht).

Register Name	Adresse	Wert
ODCNTL	0x21	0x06

Schreibe 0xE0 in das Register CNTL1 um den Sensor einzuschalten (PC 1 = 1). Die weiteren eingestellten Optionen sind dabei: Power Mode (RES = 1), Data Ready aktiviert (DRDYE = 1), Auflösung = ±8g (GSEL = 0).

Register Name	Adresse	Wert
CNTL1	0x1B	0xE0

Die Sensordaten können nun synchron ausgelesen werden, sobald eine steigende Flanke von INT1 detektiert wird.

1.4.3 Synchrones Auslesen ohne Hardware Interrupt

Schreibe 0x00 in das Register CNTL1 um den Sensor in den Standby-Modus zu versetzen.

Register Name	Adresse	Wert
CNTL1	0x1B	0x00

Schreibe 0x06 in das Register ODCNTL (Output Data Control) um die Ausgaberate ODR der Ausgangsdaten auf 50 Hz zu setzen (Dieser Schritt ist optional da dies den Standardeinstellungen entspricht).

Register Name	Adresse	Wert
ODCNTL	0x21	0x06

Schreibe 0xE0 in das Register CNTL1 um den Sensor einzuschalten (PC 1 = 1). Die weiteren eingestellten Optionen sind dabei: Power Mode (RES = 1), Data Ready aktiviert (DRDYE = 1), Auflösung = ±8g (GSEL = 0).

Register Name	Adresse	Wert
CNTL1	0x1B	0xE0

Die Sensordaten können nun synchron ausgelesen werden, sobald das DRDY Bit gesetzt und das Register INS2 (Interrupt Status 2) somit den Wert 0x10 angenommen hat.

Register Name	Adresse	Wert
INS2	0x17	0x10

1.4.4 Auslesen der Daten aus dem Sample Buffer

Schreibe 0x00 in das Register CNTL1 um den Sensor in den Standby-Modus zu versetzen.

Register Name	Adresse	Wert
CNTL1	0x1B	0x00

Schreibe 0x06 in das Register ODCNTL (Output Data Control) um die Ausgaberate ODR der Ausgangsdaten auf 50 Hz zu setzen (Dieser Schritt ist optional da dies den Standardeinstellungen entspricht).

Register Name	Adresse	Wert
ODCNTL	0x21	0x06

Schreibe 0x30 in das Register INC1 (Interrupt Control 1) um den Pin INT1 (Physical Interrupt) zu aktivieren, der Pin ist ein „high active“ Eingang.

Register Name	Adresse	Wert
INC1	0x22	0x30

Schreibe 0x40 in das Register INC4 (Interrupt Control 4), damit der „Buffer Full Interrupt“ am Pin INT1 ausgegeben wird.

Register Name	Adresse	Wert
INC4	0x25	0x40

Schreibe 0xE0 in das Register BUF_CNTL2 (Buffer Control 2) um den Sample Buffer zu aktivieren (BUFE = 1), die Auflösung der Beschleunigungsdaten auf 16-Bit zu setzen (BRES=1), den „Buffer Full Interrupt“ zu aktivieren (BFIE=1), und den Modus auf „Sample Buffer to FIFO“ zu setzen (BM=0).

Register Name	Adresse	Wert
BUF_CNTL2	0x5F	0xE0

Schreibe 0xE0 in das Register CNTL1 um den Sensor einzuschalten (PC 1 = 1). Die weiteren eingestellten Optionen sind dabei: Power Mode (RES = 1), Data Ready aktiviert (DRDYE = 1), Auflösung = $\pm$8g (GSEL = 0).

Register Name	Adresse	Wert
CNTL1	0x1B	0xE0

Sobald der Buffer voll ist und der Pin INT1 durch den „Buffer Full Interrupt“ auf High gezogen wird, können die Beschleunigungsdaten aus den entsprechenden Bufferregistern (Adresse 0x63 ff.) ausgelesen werden. Der Buffer enthält 516 Byte an Daten, welche bei einer 16-Bit Auflösung 86 verschiedenen Datentrippeln entspricht (Bei einer 8-Bit Auflösung können 171 Datentrippel, beziehungsweise 513 Byte an Daten gesammelt werden).

2 Aufbau des Demonstrators

In diesem Kapitel wird der Versuchsaufbau sowie die Einbindung des Demonstrators in den vorliegenden Quellcode der Computed Order Tracking Anwendung erläutert. Hierzu werden zunächst alle genutzten Geräte sowie die Verdrahtung dieser in Abschnitt 2.1 beschrieben. Darauf folgend wird auf die durchgeführten Änderungen der Programmstruktur in Abschnitt 2.2 eingegangen. Das Kapitel wird abgeschlossen in Abschnitt 2.3 mit der sensorspezifischen Terminologie, diversen Anmerkungen zu dem beobachteten Programmverhalten und sonstigen Besonderheiten, auf die während der Nutzung des Demonstrators geachtet werden sollte.

2.1 Geräte und Verdrahtung

Für die Kommunikation mit dem FusionG6 Prozessor wird ein Arduino MKR Zero genutzt. Dieser wird zur sowohl zur Programmierung des Prozessors, als auch zur Simulation des Sensorinterfaces und dem Auslesen der Prozessorausgabedaten via I^2C verwendet. Die Beschleunigungsdaten werden über den in Abschnitt 1 beschriebenen Beschleunigungssensor KX134-1211 mittels I^2C dem Prozessor zur Verfügung gestellt.

Entsprechend der in Abschnitt 1.1 beschriebenen Pinbelegung werden die Anschlüsse wie in der folgenden Tabelle 2.1 verdrahtet.

Tabelle 2.1: Verwendete Pinbelegung des Demonstrators

Pin	1,2,6,9	8,10	5	7
Anschluss	VDD	GND	SCL	SDA

2.2 Programmstruktur

Zur Implementierung des Demonstrators wurde der gegebene Quellcode angepasst und um die Initialisierung und Nutzung des KX134-1211 Sensors erweitert. Zur Einbindung des Sensors, muss dieser über seine spezifische Slave-Adresse angesprochen werden. Diese setzt sich aus mehreren Teilen zusammen und wird zunächst im folgenden Abschnitt genauer erläutert.

2.2.1 Slave Adresse des KX134-1211:

Die Slave Adresse des Sensors setzt sich zusammen aus drei Teilen: einem nutzerprogrammierbarem Teil, einem werksprogrammierten Teil und einem festen Teil. Dies erlaubt es mehrere Sensoren an den gleichen I^2C Bus anzuschließen [4, p.18]. Die Slave Adresse des KX134-1211 ist:

00111YX

X ist dabei das nutzerprogrammierbare Bit und wird gesetzt über Pin 9 (SDO/ADDR). In dem hier verwendeten Aufbau wird ADDR mit VDD verbunden wodurch X=1 ist.

Y ist das werksprogrammierte Bit. Im Falle des vorliegenden KX134-1211 ist es fix auf den Wert Y=1 gesetzt.

Somit ist Slave Adresse des Sensors in diesem Demonstratoraufbau 0011111 (im Code als 0x1F bezeichnet).

2.2.2 Änderungen der Main-Funktion:

Sowohl das Einlesen der Daten, als auch das Einlesen der Konfigurationsdatei wurden entfernt. Die Konfigurationsdatei nun als fester Bestandteil des Codes vorhanden, das Einlesen wurde innerhalb der Datenverarbeitung eingefügt. Die entsprechenden Konfigurationseinstellungen können der folgenden Tabelle 2.2 entnommen werden.

Tabelle 2.2: Einstellungsparameter der festcodierten Konfiguration

Variablenname	Wertebereich	Beschreibung
Direction	1,-1	Drehrichtung des Sensors
Do_order_analysis	Bool	Ein- /Ausschalten der Ordnungsanalyse
Do_envelope_detection	Bool	Ein- /Ausschalten der Hüllkurvendetektion
Sample_rate	Int	Mittlere Samplerate
Sample_rate_max	Int	Maximale Samplerate
Bandpass_low	Float	Untere Grenzfrequenz des Bandpassfilters
Bandpass_high	Float	Obere Grenzfrequenz des Bandpassfilters
FFT_size	Int	Fenstergröße der FFT
Frequency_resolution	Int	Frequenzauflösung der Berechnung
Rotation_speed_min	Int	Minimale Umdrehungszahl
Rotation_speed_max	Int	Maximale Umdrehungszahl
Divisions	Int	Anzahl der betrachteten Frequenzordnungen
Rotations	Int	Anzahl der Rotationen die ein FFT-Fenster beinhaltet

Um die Beschleunigungsdaten des Sensors auslesen zu können, muss dieser zunächst mittels I²C initialisiert werden. Dies soll anhand des folgenden Codeabschnitts beispielhaft gezeigt werden.

Algorithm 1 Initialisierung des Beschleunigungssensors

1: $uint8\_t \quad setup\_var[2] = 0x1B, 0x00$
2: $i2c\_send(0x1F, \&setup\_var, 2)$

Entsprechend der in Abschnitt 1 beschriebenen Register und Kommunikationsprotokolle wird über setup_var[0] angegeben welches Register beschrieben werden soll, in setup_var[1] befindet sich der zu schreibende Registerinhalt. Über den Befehl i2c_send wird der Sensor angesprochen. Hierbei beinhaltet das erste Argument die Slave-Adresse des Sensors, welche in diesem Demonstratoraufbau auf 0x1F gesetzt wurde. In Argument 2 wird der Pointer des Initialisierungarrays übergeben, in Argument 3 wird die Anzahl der zu sendenden Pakete beschrieben. Der Sensor wird in diesem Demonstratoraufbau mit folgenden Optionen initialisiert:

- High-Performance Modus : Niedrigeres Rauschen auf den Beschleunigungsdaten
- "Data Ready Engine" deaktiviert, ermöglicht das asynchrone Auslesen der Datenregister
- Auflösung der Beschleunigungsdaten $= \pm 8$g

Anschließend werden die Sensordaten, hier die Beschleunigungsdaten in X-Richtung, ausgelesen. Die Werte werden, entsprechend der in Abschnitt 1.2 gegebenen Tabelle, aus den Registern 8 und 9 ausgelesen und zu einem 16 Bit Beschleunigungswert zusammengerechnet. Da die Beschleunigung jedoch als Wert zwischen 32767 und -32768 ausgegeben wird, müssen die ausgelesenen Daten zunächst umgewandelt werden. Dies wird über die folgende Gleichung durchgeführt:

$$Acceleration = ((X\_OUT_H << 8 + X\_OUT_L)/32767) * 8 * 8.91 \tag{2.1}$$

Darauffolgend ist eine Laufschleife eingefügt, welche für diesen Demonstratoraufbau Dummy-Positionsdaten generiert. Nach der Initialisierung des Gesamtsystems werden solange Daten eingelesen bis ein Datenblock prozessiert werden kann (512 Datenpaare). Abschließend wird das Filtersystem nach jeder durchgeführten FFT mittels eines Soft-Resets zurückgesetzt. Dies wird benötigt, da sonst nach jedem prozessierten Datenblock eine FFT durchgeführt und dieselben Daten mehrfach verarbeitet werden würden.

2.2.3 Änderungen der Process Block-Funktion:

Da der Sensor im Falle des Demonstrators keine frequenzbehaftete Beschleunigung erfährt, muss der Bandpass in diesem Aufbau überbrückt werden da ansonsten sämtliche Beschleunigungdaten genullt werden würden.

2.3 Anmerkungen

Kraft g:

Der Sensor wird Initialisiert und gibt die Daten in Abhängigkeit der Erdanziehungskraft an. Hierbei gilt:

$$1g = 9.8\,\mathrm{m/s^2} \tag{2.2}$$

Ein Tausendstel g ($0.0098\,\mathrm{m/s^2}$) wird dabei als 1 milli-g (1 mg) bezeichnet.

Zero-g Offset:

Werden Daten ausgelesen während der Sensor keine Beschleunigung erfährt, enthalten die Register OUTX, OUTY und OUTZ idealerweise den Wert null. Dieser entspricht dem Mittelwert des dynamischen Wertebereiches des Sensors. Durch verschiedene Einflüsse (Missmatch bei der Fertigung, Fehler/Ungenauigkeiten bei der Kalibrierung, Mechanischer Stress) können die Sensordaten von diesem idealen Ausgangswert abweichen. Diese Abweichungen werden als „Zero-g Offset“ bezeichnet [4, p.10].

I²C Interface:

Das I²C Interface des Sensors unterstütz das Standard-I²C Kommunikationsprotokoll, sowie das „Fast Mode“ und „High-Speed Mode“ Kommunikationsprotokoll.

Limitierung der Eingangsdaten:

Entsprechend der Anmerkung in [5, p.34 ff.] muss bei den Positionsdaten (X-Wert der Eingangsdaten) ein eingeschränkter Wertebereich eingehalten werden, da das Programm sonst keine korrekten Ergebnisse berechnet.

Selbiges gilt für die Beschleunigungsdaten (Y-Wert der Eingangsdaten). Sind diese zu klein, werden durch einen Fehler bei der Berechnung der Ringbufferpointer in der Interpolation die anschließenden Programmschritte nicht mehr durchgeführt (zweite Dezimierung sowie FFT). Um diesen Fall abzufangen sollte, wenn möglich, eventuell ein Abfangmechanismus im Code eingebaut werden der entweder die Ringbufferpointer im Anschluss an die Interpolation korrekt berechnet. Alternativ könnten die Interpolationsdaten in eine neue Ringbufferstruktur geschrieben werden.

Zurücksetzen der I²C Kommunikation:

Darauf folgend muss die I²C Kommunikation zurückgesetzt werden, da diese sonst aufgrund eines Bugs die LSBs der empfangenen Daten durch ein frühzeitiges Stop-Signal abschneidet.

Literaturverzeichnis

[1] "AN113 transitioning to KX134-1211 accelerometer." [Online]. Available: https://kionixfs.azureedge.net/en/document/AN113-Transitioning-to-KX134-1211-Accelerometer.pdf

[2] I. Kinoxi, "KX134-1211-technical-reference-manual-rev-3.0.pdf." [Online]. Available: https://kionixfs.azureedge.net/en/document/KX134-1211-Technical-Reference-Manual-Rev-3.0.pdf

[3] ——, "AN101 getting started." [Online]. Available: https://kionixfs.azureedge.net/en/document/AN101-Getting-Started.pdf

[4] ——, "KX134-1211-specifications-rev-1.0-1659717.pdf." [Online]. Available: https://www.mouser.de/datasheet/2/348/KX134-1211-Specifications-Rev-1.0-1659717.pdf

[5] J. Rinke, "Evaluation and optimization of a complex filter system using different application-specific instruction-set processor configurations."