



FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT



Lehrstuhl für Technische Elektronik

Prof. Dr.-Ing. Dr.-Ing. habil. Robert Weigel

Prof. Dr.-Ing. Georg Fischer

Bachelorarbeit

im Studiengang

„Elektrotechnik, Elektronik und Informationstechnik (EEI)“

von

Christof Pfannenmüller

zum Thema

Aufbau und Inbetriebnahme einer mobilen Basisstation für feldstärkebasierte Lokalisierung

Betreuer: Dipl.-Ing. Felix Pflaum

Beginn: 25.04.2016

Abgabe: 26.09.2016

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde.

Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, den 26. September 2016

Christof Pfannenmüller

Kurzfassung

Ziel der vorliegenden Arbeit war der Aufbau einer Basisstation mit sechs unabhängigen Transceivern. Der Einsatz dieses Aufbaus sollte der späteren Lokalisierung von Sensoren oder anderen Sendern im Sub-GHz-Frequenzbereich um 868 MHz dienen. Die relative Ortsbestimmung zur Basis sollte energieeffizient sein und gleichzeitig eine hohe Auflösung bieten. Dazu sollte die genaue Position des Senders auf Basis der unterschiedlichen Feldstärken an den Transceivern eruiert werden. Ausgenutzt werden sollte dabei, dass typische integrierte Transceiver die Empfangsfeldstärke selbst auswerten und diese ausgelesen werden kann. Das Erkennen von Übertragungen und das anschließende Auslesen der ankommenden Daten sollte dabei ein Mikrocontroller übernehmen. Zur weiteren Verarbeitung der Daten sollten diese anschließend einem Computer zur Verfügung gestellt werden. Dazu wurde sowohl eine USB- als auch eine Netzwerk-Schnittstelle vorgesehen. Die beim Aufbau der Platine verwendete Hardware basierte zum Großteil auf Bauteilen des Herstellers Infineon. Beim Layout der Platine wurden die sechs Transceiver sternförmig und regelmäßig um den Mikrocontroller und dessen Peripherie angeordnet, um ein gleichmäßiges Empfangsverhalten aus allen Raumrichtungen zu gewähren. Die Funksegmente der Platine wurden so gestaltet, dass diese bei Bedarf abgetrennt und mit einer Kabelverbindung weiter voneinander entfernt werden konnten. Antennen zum Senden sollten über Steckverbinder an die Basisstation angeschlossen werden. Der anschließende Softwareentwurf für die Basisstation nutzte zu einem Großteil bereits bestehende Bibliotheken und sollte ankommende Übertragungen erkennen, die zur Ortung notwendigen gemessenen Werte abfragen und an den Hostcomputer weiterleiten.

Abstract

Current localization measurements by radio frequency have been complex and consume plenty of energy. However almost every Receiver has knowledge of the electrical field strength, correlating to distance from Transmitter, this information is nearly unused. A multi Transceiver base station should start communication with a mobile wireless sensor. The relative positioning to the base could be calculated by the received signal strength (RSSI) already provided from Transceivers without additional components. Due to the permeability of walls, the possible range and the wavelength, associated to resolution of localization, Sub-GHz frequency range is used. Therefoer six identical Transceiver-ICs were arranged over all horizontal directions in space. By use of focused antenna the recognition of transmission direction could be corrected. Coordination of the ICs would be realized by a XMC4500 μ C connected to the Transceivers with SPI and IRQ line for finished transmission. Distribution of received data and signal strength measurements to a host computer is accomplished by the XMC4500 over Ethernet and USB.

Abkürzungsverzeichnis

PCB	Printed Circuit Board	5
EDA	Electronic Design Automation	5
CAD	Computer-aided design	5
SMD	Surface-mounted device	6
SPI	Serial Peripheral Interface	7
DRC	Design-Rule-Check	7
FIFO	First In – First Out	8
IC	Integrierter Schaltkreis	9
NCS	Non-Chip-Select	8
SMA	Sub-Miniature-A	9
LQFP	Low Profile Quad Flat Package	10
BGA	Ball Grid Array	10
JTAG	Joint Test Action Group	11
TVS	Transient Voltage Suppressor	11
LDO	Low Drop-Out	12
SOT	Small Outline Transistor	12
STEP	Standard for the Exchange of Product Model Data	13
NC	Numerical Control	13
IDE	integrated development environment	15
GUI	Graphical User Interface	15
SDK	Software development kit	15
CPU	Central Processing Unit	16
USIC	Universal Serial Interface Channel	16
ETH	Ethernet MAC (Ethernet Medium Access Control)	16
USB	Universal Serial Bus	16
GPIO	General Purpose Input/Output	16

ISR	Interrupt Service Routine	17
ERU	Event Request Unit	17
ERS	Event Request Select	17
ETL	Event Trigger Logic	17
OGU	Output Gating Unit	17
NVIC	Nested Vectored Interrupt Controller	17
CMSIS	Cortex Microcontroller Software Interface Standard	18
MISO	Master-In Slave-Out	18
MOSI	Master-Out Slave-In	18
IRQ	Interrupt Request	20
PLL	Phasenregelschleife (phase-locked loop)	20
AGC	automatic gain control	21
GND	Masse (Ground)	18
RSSI	Received Signal Strength Indication	22

Inhaltsverzeichnis

1	Einleitung	3
1.1	Motivation	3
1.2	Zieldefinition	4
1.3	Projektmanagement	4
2	Platinenaufbau	5
2.1	Vorüberlegungen	5
2.2	Layoutprogramm Altium Designer	5
2.3	verwendete Hardware	8
2.3.1	TDA5340	8
2.3.2	XMC4500	10
2.3.3	Ethernet	11
2.3.4	Spannungsversorgung	12
2.4	Generierte Dokumente	13
2.5	Bestückung	13
3	Software	15
3.1	DAVE Entwicklungsumgebung	15
3.2	verwendete Peripherie des XMC4500	16
3.2.1	GPIO	16
3.2.2	USIC	16
3.2.3	ERU	17
3.2.4	USB	17
3.2.5	Ethernet	17
3.3	verwendete Bibliotheken	18
3.3.1	XMC Library (XMC Lib)	18
3.3.2	SPI Library	18
3.3.3	TDA5340 Library	19
3.3.4	Virtueller COM Port	19
3.4	Programmablauf	20
3.4.1	Konfiguration der Funkmodule	20
3.4.2	interruptbasierte Datenerfassung	20
3.4.3	Weiterleitung der erfassten Daten	21

4	Feldtest	23
4.1	Aufbau	23
4.2	Durchführung	23
4.3	Ergebnisse und Auswertung	24
5	Zusammenfassung und Ausblick	25
	Abbildungsverzeichnis	27
	Tabellenverzeichnis	29
	Quellcodeverzeichnis	31
6	Anhang	33
6.1	Seriennummern	33
6.2	3D-Daten	33
6.2.1	Platine	33
6.2.2	Gehäuse	33
6.3	Layout TDA5340 Aufsteckboard	34
6.4	Quellcode	34
6.4.1	Main.c	34
6.4.2	ISRs.c	34
6.4.3	Init.c	36

Einleitung

1.1 Motivation

Die Lokalisierung von Objekten bildet in der Sicherheits- und Automatisierungstechnik eine immer größere Basisdisziplin. Sie ist daher seit Anbeginn des Informationszeitalters eine zentrale Aufgabe der Elektronik. Im Zeitalter von Industrie 4.0 und allgegenwärtigen autonomen Systemen gewinnt sie zunehmend an noch größerer Bedeutung.

Bisherige Ansätze zur elektronischen Ortsbestimmung sind meist aufwändig mit einem hohen Energieverbrauch und benötigen viele Teilkomponenten. Dies macht die elektronische Ortsbestimmung teuer, was sich vor allem im Betrieb von Systemen wie GPS oder Galileo zeigt. Daneben haben diese Systeme vor allem den Nachteil, dass die verwendeten Frequenzen Wände kaum durchdringen und somit innerhalb von Gebäuden nicht verwendet werden können. Genau dieses Anwendungsszenario stellt jedoch in der Industrie 4.0 eine typische Fabrik dar. Dies führt dazu, dass für das am häufigsten verwendete Beispiel, einen autonom agierenden Roboter in einer Lagerhalle, andere Techniken zur Ortsbestimmung eingesetzt werden müssen.

Ortungsansätze für ein solches autonomes bewegliches System existieren bereits und werden wie bei der WLAN-basierte Ortung auch eingesetzt. Diese Systeme haben jedoch den Nachteil, dass sie viele Sender mit bekanntem Ort benötigen. Dies erfordert eine aufwändige Kalibrierung des Systems und führt vor allem zu einem großen Energieverbrauch des Gesamtsystems, da alle stationären Einheiten dauerhaft senden müssen, um eine Ortung zu ermöglichen. Ein weiterer Nachteil der Systeme ist, dass nur der zu ortenden Client die Informationen über seinen Standort hat. Für ein außenstehendes System, wie den steuernden Hauptrechner einer Industrieanlage, ist nicht festzustellen, wo das Objekt sich befindet. Durch diese Probleme ließe sich das Anwendungsszenario eines einzelnen autarken Funksensors, dessen Messwert vom Hauptrechner abgefragt wird, nur schwer realisieren. Der Sensor müsste über eine große Energiereserve verfügen und ständig aktiv sein, um zu erkennen wenn er dazu aufgefordert wird seine Position zu ermitteln und mitzuteilen. Durch das Verlagern der Ortungsaufgabe auf das stationäre System müsste der Sensor nur beim Vorliegen eines neuen Messwertes eine Funkverbindung aufbauen. Die Basisstation könnte aus dem empfangenen Signal sowohl den Messwert extrahieren

als auch eine Lokalisierung durchführen. Sollte der stationäre Teil aus nur einer Einheit bestehen, wäre auch eine Änderung am Gesamtsystems, wie eine Ortsänderung der stationären Einheit, einfach möglich, da nur die relative Position zu dieser ermittelt wird, weshalb eine Neukalibrierung entfällt. Der Sensor selbst benötigt im Normalfall keine Information über seinen Aufenthaltsort. Dies erlaubt eine mobile Ortung da die Basis unkompliziert bewegt werden kann.

Es bietet sich hierfür eine Nutzung des Sub-GHz-Frequenzbandes an, welches deutlich weniger ausgenutzt ist, in dem meist nur kurze Funkdauern verwendet werden und in welchem eine Übertragung vor allem energieeffizienter möglich ist. Ein weiterer Vorteil des verwendeten Frequenzbandes sind die bessere Penetrationseigenschaften durch Wände und Personen im Vergleich zu Ansätzen, die etwa auf WLAN basieren.

1.2 Zieldefinition

Zum Umsetzen einer solchen oben beschriebenen Ortung sollte eine Basisstation aufgebaut werden, welche als stationäre Einheit einen Sender orten könnte. Die Aufgabenstellung dieser Bachelorarbeit bestand im Einarbeiten in den vorgegeben Transceiver TDA5340 und den Mikrocontroller XMC4500. Bei der Gestaltung des Layouts für die Platine mit Altium Designer sollten diese über einen SPI-Bus verbunden sein. Die Platine sollte zusätzlich ein Konzept zur Bereitstellung der Versorgungsspannung enthalten. Um die Weiterverarbeitung der bezogenen Messwerte sicherzustellen, sollte eine LAN-Schnittstelle sowie eine Möglichkeit für einen USB-Anschluss zu einem PC integriert werden. Das ursprünglich geplante Design von PCB-Antennen an den Transceivern auf der Platine wurde wegen des Umfangs der dazu notwendigen Simulationen zu Beginn aus der Aufgabenstellung entfernt. Stattdessen sollten zugekaufte Antennen, verbunden über Steckverbinder, mit der Basisstation verwendet werden. Eine funktionierende Firmware für das Initialisieren und Betreiben der Basisstation sollte ebenfalls mit der dazu notwendigen Entwicklungsumgebung erstellt werden. So sollte der Mikrocontroller auf, vom Transceiver ausgelöst, Interrupts für ankommende Übertragungen reagieren und Messwerte erfassen und entsprechend weiterleiten. Durch die Verwendung von Makros beim Softwareentwurf sollte eine spätere Anpassung, wie das Tauschen von Pins der leichter möglich sein.

1.3 Projektmanagement

Die vorliegende Arbeit wurde innerhalb von fünf Monaten am Lehrstuhl für Technische Elektronik der Universität Erlangen-Nürnberg angefertigt. Dabei lag der Fokus in den ersten beiden Monaten auf dem Layout der Platine und dem anschließenden Bestücken. In der folgenden Zeit wurde vermehrt auf die Software für den Betrieb der Basisstation eingegangen. Außerdem wurde in den letzten zwei Monaten die Dokumentation mit Latex erstellt. Als Versionskontrolle für das gesamte Projekt wurde Github eingesetzt.

Platinenaufbau

2.1 Vorüberlegungen

Die Zielsetzung im Aufbau der Platine war eine kompakte Basisstation mit sechs Transceivern und einer zentralen Steuereinheit. Um sicherzustellen, dass alle Antennen gleichmäßig in die sechs vorgegebenen Raumrichtungen abstrahlen, sollte bereits die Platine symmetrisch aufgebaut werden. Dazu wurde zuerst das Layout der sechs identischen Transceiver-Einheiten mit dem TDA5340 Baustein und den Antennenanschlüssen erstellt und anschließend gleichmäßig um die weiteren für die Schaltung notwendigen funktionellen Segmente angeordnet.

2.2 Layoutprogramm Altium Designer

Bei dem Entwicklungswerkzeug „Altium Designer“ des Entwicklers Altium Limited handelt es sich um ein System zum Entwurf von gedruckten Schaltungen oder Printed Circuit Boards (PCBs). Ein solches Programm wird auch als Electronic Design Automation (EDA) oder ECAD für electronic Computer-aided design (CAD) bezeichnet, da es den Entwickler bei der Umsetzung der Anforderungen in einen Schaltplan und später eine Platine unterstützen soll. Wie viele andere EDA-Programme ist auch Altium Designer so aufgebaut, dass sich der Entwickler zuerst mit dem allgemeinen symbolisierten Schaltplan befassen kann und erst zu einem späteren Zeitpunkt die tatsächliche Anordnung der Bauteile auf dem PCB-Substrat festgelegt wird. Somit können zuerst im Schematic Editor die Funktionen der Schaltung umgesetzt werden. Dazu werden die verwendeten Bauteile aus zuvor angelegten Bibliotheken verwendet oder es werden bestehende Bibliotheken verwendet, die etwa vom Hersteller der Bauteile zur Verfügung gestellt werden. Altium selbst bietet hierfür auch diverse Möglichkeiten an und stellt Bauteile nach Hersteller und Art geordnet bereit. In den Bibliotheken sind alle im weiteren Verlauf benötigten Informationen über die einzelnen Bauteile enthalten. So liegen dort etwa entsprechenden Abbildungen für das Bauteil im Schaltplan vor. In den so genannten „Footprints“ zu jedem Bauteil, welche ebenfalls in den Bibliotheken enthalten sind, wurde zuvor die, für das

physikalische Gehäuse, notwendigen Abmessungen, Löt pads und Abmessungen für Lötstopplack um das Bauteil festgelegt. Da es Bauteile, wie den verwendeten Mikrocontroller, in verschiedenen Gehäusen geben kann, besteht somit auch die Möglichkeit hier zwischen verschiedenen Footprints zu wählen. Da viele Gehäuse herstellerübergreifend genormt sind, konnten teilweise bestehende Footprints genutzt oder diese mehrfach verwendet werden.

Wie bereits erwähnt wird im EDA-Programm zuerst der symbolische Schaltplan erstellt. Dieser wird anschließend in ein Layout für eine Platine umgewandelt. Die zu den Schaltplansymbolen korrespondierenden Footprints werden dazu auf dem Layout der Platine angeordnet und durch das „Routing“ werden die Leiterbahnen definiert. Altium Designer ist dabei in drei Teilbereiche unterteilt: im „Board Planning Mode“ liegt der Fokus auf dem Anordnen der einzelnen Bauteile und Komponenten auf der Leiterplatte, außerdem wird in diesem Bereich die Form und das Ausmaß der Leiterplatte festgelegt. Im 2D-Modus des PCB-Editor lassen sich anschließend die aus der Definition im Schaltplan ergebenden elektrischen Verbindungen örtlich auf den verschiedenen Kupferebenen (Layern) anordnen. Die Hauptarbeit findet also in diesem Teil des PCB-Editors statt. Der 3D-Modus dient anschließend zur Evaluation des Designs und zur Anpassung an Gehäuse oder andere Komponenten. In den verschiedenen Ebenen oder „Layern“ sind die Kupferebenen und andere Schichten der späteren Platine wie der Bestückungsdruck oder der Lötstopplack gesammelt. Jede Ebene entspricht daher einer zu fertigenden Schicht und existiert für die Vorder- und Rückseite der Platine. Beim Bewegen eines Bauteils wird nicht nur die markierte Abmessung, sondern etwa auch die Anschluss pads und der Lötstopplack auf denen entsprechenden Ebenen bewegt. Durch Vias sind elektrische Verbindungen zwischen den Kupferschichten möglich.

Für die Basisstation wurden zwei Kupferebenen und ausschließlich sogenannte Surface-mounted device (SMD)-Bauelemente verwendet. Diese liegen nur auf der Oberfläche der Platine auf und sind durch ihre Lötverbindungen befestigt. Die Vorteile dieser Bauteile sind der geringe Preis und die kleinen Abmessungen.

Wegen der Größe des Projekts wurde zur besseren Übersicht ein so genanntes „Multi-Sheet-Design“ erstellt. Dadurch war es möglich, die verschiedenen funktionellen Blöcke der Basisstation auf getrennte Blätter des Schaltplans zu verteilen. Der Mikrocontroller, die Spannungsversorgung und der Transceiver, sowie die für eine Netzkommunikation notwendigen Bauteile wurden dabei auf unabhängigen Seiten angeordnet und dort die elektrischen Verbindungen erstellt.

Da der Transceiver und die entsprechende Peripherie sechs mal in identischer Anordnung und Beschaltung verwendet wurden und auch auf dem PCB-Substrat mehrfach mit Leiterbahnen verbunden und angeordnet werden mussten, wurde hierfür ein so genanntes „Multi-Channel“-Design gewählt. In Altium Designer können mit diesem Feature identische Schaltungsteile einmal angeordnet, mit Leiterbahnen verbunden und dieses Design auf alle anderen entsprechenden Schaltungsteile angewendet werden. Somit muss das aufwändige Anordnen der Bauteile und die Führung der Leiterbahnen nur bei einem der Kanäle durchgeführt werden. Dazu wurde mit Hilfe eines übergeordneten Sheet-Symbols der Schaltplan des Transceivers in den Schaltplan des Mikrocontroller eingefügt und diesem somit untergeordnet. Über Ports, welche zum Schaltplansymbol hinzugefügt werden, lassen sich elektrische Verbindungen zwischen Netzen innerhalb der Schaltpläne

für Transceiver und Mikrocontroller erstellen. Dabei wird im Transceiver-Schaltplan ein Port hinzugefügt, der mit dem gewünschten elektrischen Netz verbunden werden kann. Auf dem Schaltplan-Symbol im Mikrocontroller-Schaltplan wird ein entsprechender gleichnamiger Port erstellt, der mit Netzen am Mikrocontroller verbunden werden kann. Da die drei Verbindungen der Serial Peripheral Interface (SPI)-Kommunikation jeweils aus einem Netz bestehen und etwa alle MISO-Leitungen an demselben Pin des XMC4500 und denselben Anschluss bei allen TDAs angebunden sind, konnte hierfür ein einfacher Port verwendet werden. Alle anderen Anschlüsse, wie etwa die Auswahlleitung für die SPI-Verbindung, welche für jede der sechs verschiedenen Transceiver-Einheiten mit einem anderen Anschluss des Mikrocontrollers verbunden sein mussten, wurden deswegen mit dem Repeat-Kommando erstellt. So wird im untergeordneten Schaltplan, in diesem Fall dem des TDA, der Port beliebig benannt, etwa als „NCS“. Der auf dem Schaltplansymbol erstellte korrespondierende Port wird dagegen in „Repeat(NCS)“ umbenannt. Eine Verbindung mit dem Port auf dem Schaltplansymbol wird dadurch zu einem Bus. Dieser kann aufgetrennt werden und die Verbindungen von jedem Kanal als einzelnen Signal an den Mikrocontroller angeschlossen werden. Globale Netze wie die Versorgungsspannung von 3,3 Volt oder die Masse müssen dabei nicht als Port hinzugefügt werden. Altium Designer stellt deren Verbindungen automatisch her. Wird nun noch der Name des Schaltplansymbols nach dem folgenden Muster angepasst, wird Altium Designer beim Kompilieren des Projektes ein Multi-Channel-Design erstellen, die Kanäle wie angegeben durchnummerieren und den Kanal im PCB-Editor entsprechend mehrfach erstellen. Dazu sollte der Name die Struktur „Repeat(<Name>,<Startnummer> ,<Endnummer>)“ aufweisen. Anschließend kann die Anordnung und das Routing des ersten Kanals erfolgen. Da für jeden Kanal ein eigener so genannter „Room“ erstellt wird, lassen sich, nach dem Erstellen der Leiterbahnen des ersten Kanals, die Anordnungen der Bauteile und Leiterbahnen mit dem „Copy Room Format“-Befehl auf alle anderen Kanäle erweitern. Ein Room bezeichnet dabei eine Gruppe an Elementen im PCB-Editor und einen Bereich der Platine, in dem diese angeordnet sind und erlaubt so das Verschieben ganzer Schaltungsteile auf der Platine.

Altium Designer nummeriert automatisch die verwendeten Bauelemente durch, um die genaue Identifikation eines Bauteiles zu erlauben und erstellt automatisch einen Aufdruck neben dem Lötpad mit dem Bauteilnamen in der entsprechenden Ebene. Ein solcher Bestückungsdruck zur leichteren Anordnung der Bauteile auf der fertig entwickelten Platine wurde nur auf der Hauptplatine erstellt. Auf den Teilplatinen für die Transceiver wurde dieser aus Platzgründen weggelassen.

Um sicherzustellen, dass die sich aus der Bauteilanordnung ergebenden Pads und die Leiterbahnen auch fertigbar sind, stellt Altium Designer zwei Design-Rule-Checks (DRCs) bereit. Im ersten Live-DRC werden „Violations“, also Bauteile mit Verstößen gegen die Designregeln, durch ein farbiges Overlay markiert. Im folgenden kompletten Test zeigt Altium alle weiteren Verstöße an. Als Verstoß gelten etwa Leiterbahnen unterschiedlicher Netze, die sich berühren und zu einem Kurzschluss führen würden oder auch Verletzungen der festgelegten Abstandsregeln (Clearance). Alle Regeln für Violations können im „PCB Rules Editor“ eingestellt werden und so an die Möglichkeiten des PCB-Herstellers angepasst werden. Da die Leiterplatte für die Basisstation von Multi Circuit Boards Ltd. hergestellt wurde, konnten die auf der Webseite dieses Herstellers angegebenen

Designregeln für den DRC übernommen werden. Da Altium Designer im DRC sich kurzgeschlossene und falsch verbundene Pads angezeigt werden, muss kein extra „Layout vs. Schematic“-Test durchgeführt werden, der den vorher erstellten Schaltplan mit dem Layout abgleicht. Solche Unterschiede zwischen Schaltplan und dem PCB-Layout würden durch den DRC bereits angezeigt.

2.3 verwendete Hardware

2.3.1 TDA5340

Der verwendete Transceiver TDA5340 wird von Infineon Technologies AG entwickelt und vertrieben. Er ist Teil der SmartLEWISTM Produktfamilie die energiesparende Lösungen für Funkanwendungen im Frequenzspektrum unterhalb von einem Gigahertz bietet. Der Transceiver kommuniziert mit seinem Host über das SPI-Protokoll, der Mikrocontroller ist in diesem Fall sternförmig mit den einzelnen TDA-Bausteinen verbunden, die als Slaves fungieren. Die Daten werden mit drei gemeinsamen Leitungen übertragen, eine vierte Leitung dient dem XMC zur Auswahl des gewünschten Slaves für die Kommunikation. Diese Non-Chip-Select (NCS)-Leitung arbeitet active-low, sodass der jeweilige TDA5340 eine Interaktion akzeptiert, sobald diese vom XMC-Baustein auf Massepotential gezogen wird. Von den drei eigentlichen Datenleitungen fungiert eine als reiner Ausgang des Masters bzw. Dateneingang des TDA (MOSI), eine zweite als Eingang des Masters (MISO) und die dritte als ein vom Mikrocontroller getriebenes Clock-Signal. Bei dem auf MISO und MOSI anliegenden Signal handelt es sich um ein unipolar kodierte non-return-to-zero Signal, welches einer logischen 0 bei Massepotential entspricht. Der TDA unterstützt acht verschiedene Instruktionen, die es erlauben entweder einzelne Register des Bausteins zu lesen bzw. zu schreiben, auf mehrere hintereinander folgende Register oder auf die beiden Puffer des Bausteins zuzugreifen. In den beiden Puffern, die als First In – First Out (FIFO)-Strukturen aufgebaut sind, werden die vom TDA erkannten und demodulierten bzw. die auf Übertragung wartenden Signalkomplexe zwischengespeichert. Diese Zwischenspeicherung soll den Mikrocontroller entlasten, so können entsprechende Datenpakete dem TDA5340 mitgeteilt werden und dieser übernimmt selbsttätig eine korrekte Modulation und Übertragung mit den eingestellten Parametern.

Der TDA5340 kann sowohl mit einer Spannungsversorgungsspannung von 5V als auch bei 3,3V arbeiten. Da aber der XMC nur bei letzterer betrieben werden kann, wurde der TDA-Baustein und die externe Beschaltung einfachheitshalber auch auf 3,3V ausgelegt. Um zu einem späterem Zeitpunkt eine größere Entfernung zwischen den einzelnen Antennen, und somit auch den jeweiligen Transceivern zu erlauben, wurde eine Sollbruchstelle vorgesehen. Dadurch könnten die gesamte Baugruppe von der Mutterplatine entfernt werden, was unter Umständen notwendig gewesen wäre, um größere Unterschiede in der Signalstärke an den einzelnen Transceivern zu erhalten und somit eine bessere Auflösung in der Ortung zu erlauben. Dazu wurden Anschlussleisten im Rastermaß 2,54mm an beiden Seiten der Sollbruchstelle vorgesehen. Die Verbindung der Transceiver-Einheiten mit der Hauptplatine wurde über diese Sollbruchstelle hinweg mit Leiterbahnen gewährleistet. Nach dem Abtrennen der TDA-Teilplatine, an der durch Bohrungen vorgesehenen Bruchstelle, wäre die elektrische Verbindung durch Kabel sichergestellt worden. Da es

sich bei den zu übertragenden ausschließlich um digitale Signale handelt, hätte dies unproblematisch mit ungeschirmten Flachbandkabeln ermöglicht werden können. Ein solches Vergrößern des Abstandes der Antennen war jedoch nicht notwendig. Neben der Versorgungsspannung, Masse und den vier für die SPI-Kommunikation notwendigen Signalen wurden noch die drei multifunktionalen Digitalausgänge und der Power-On Reset-Pin (P_ON) dem XMC4500 über die Buchsenleisten zur Verfügung gestellt.

Die drei multifunktionalen Digitalausgänge PP0, PP1 und PP2 des Transceivers können durch entsprechende Kalibrierung von Registers im Integrierter Schaltkreis (IC) mit verschiedenen Signalen belegt werden. So ist es etwa möglich mit diesen einen externen Antennenumschalter zu steuern oder andere Signale wie ein Clocksignal auszugeben. PP2 wird vom TDA standardmäßig als Interrupt Signal verwendet. Die nicht verwendeten PP1 und PP0 wurden bei der späteren Konfiguration in den hochohmige Zustand geschaltet. Durch einen am P_ON-Pin anliegenden High-Pegel wechseln der Transceiver in den Schlafmodus. Anderenfalls ist dieser ausgeschaltet.

Die Antenne wurde am oberen Ende jeder TDA5340-Teilplatine vorgesehen. Als Anschluss für die Antenne wurde hier eine Koaxialbuchse in Sub-Miniature-A (SMA)-Ausführung verwendet, welche auf 50Ω angepasst ist. Durch den Koaxialsteckverbinder konnte sichergestellt werden, dass alle notwendigen Frequenzen auch korrekt und unge-dämpft passieren können. Das Anpassnetzwerk zwischen dem integrierten Transceiver und der verwendeten SMA-Buchse diente der Leistungsanpassung zwischen den Pins des TDA5340 und der 50Ω -Koaxialbuchse. Der Aufbau des Anpassnetzwerkes basiert auf einem von Stefan Erhard erstellten Schaltplan für eine Aufsteckplatine für das Evaluationsboard „XMC 2Go“ von Infineon. Durch die Verwendung von hoch abgestimmten Spulen und Kondensatoren mit Toleranzen von nur $\pm 0,05\text{pF}$ bei einem Nennwert von $2,5\text{pF}$ wurde die korrekte Anpassung sichergestellt.

Zur Verbesserung der Hochfrequenzeigenschaften wurden die Freiräume zwischen den Leiterbahnen mit einer Kupferfläche gefüllt, die mit dem Masseanschluss kontaktiert war. Durch die Verwendung von Vias, vor allem im Bereich des Anpassnetzwerkes, sollte eine niederohmige Verbindung zwischen den beiden Masseflächen auf der Ober- bzw. Unterseite der Platine erreicht werden. Daneben dienten diese, auf Nullpotential liegenden Vias, jedoch vor allem der Abschirmung der Pfade für die HF-Signale gegen mögliche Einkoppelungen aus der Umgebung, welche ankommende Funksignale stören könnten.

Obwohl der TDA5340 einen eingebauten Zwischenfrequenz-Filter hat, der über eine umschaltbare Bandweite verfügt, wurde ein externer Keramikfilter verwendet. Der TDA stellt dafür zwei Pins bereit, zwischen denen ein solcher Filter mit einer Frequenz von $10,7\text{ MHz}$ angeschlossen werden kann. Ohne einen hier extern angeschlossenen Filter würde der TDA als einfacher heterodyner Mischer direkt auf die Zwischenfrequenz 274 kHz heruntermischen. Bei Verwendung eines externen Keramik oder auch eines LC- π -Filters kann das ankommende HF-Signal jedoch in zwei Stufen gefiltert werden, ehe es in das Basisband demoduliert wird, was zu einer höheren Signalqualität führt. Die Umstellung zwischen einfacher und Double Down Conversion erfolgt durch das setzen eines dafür vorgesehenen Bits

Die elektrischen Verbindungen zwischen den beiden Eingängen des Low Noise Amplifier und dem Anpassnetzwerk wurden mit dem „Differential Pair Routing“-Feature von Altium Designer erstellt. Durch einen im Schaltplan auf die positive und negative elektrische

Leitung zwischen dem TDA5340 und dem Anpassnetzwerk angewendeten Parameter wird das Leitungspaar als differentiell markiert. Anschließend kann mit dem interaktiven „Differential Pair Routing“ einer der beiden Leitungen begonnen werden. Altium Designer wird dabei selbstständig versuchen, die zweite Leiterbahn des Paares so anzuordnen, das die beiden Leiterbahnen symmetrisch und parallel zueinander liegen. Durch die Verwendung des „Differential Pair Routing“ versucht Altium Designer auch die Länge der beiden Verbindungen anzugleichen. Durch die Anpassungen der differentiellen Leiterbahnen wird eine gleichmäßige Übertragung sichergestellt. Da diese Leitungen hochfrequente Signale führen, ist eine genaue Anpassung notwendig.

Um Einkopplungen auf die Pfade für hochfrequente Signale zu vermeiden, wurde das für den Transceiver benötigte Quarz möglichst weit vom Sende- bzw. von den Empfangsanschlüssen des TDA angeordnet. Aus diesem Grund wurde der für den Oszillator benötigte Quarz mit einer Frequenz $f_{Crystal} = 21,948717 \text{ MHz}$ zwischen dem IC und der vorgesehenen Stiftleiste platziert. Die Frequenz des benötigten Quarzes ergibt sich aus dem Zusammenhang

$$f_{Crystal} = f_{IF2} * 80 = \frac{f_{IF1}}{39} * 80 = \frac{10,7 \text{ MHz}}{39} * 80 = 21,948717 \text{ MHz} \quad (2.1)$$

wobei die Zwischenfrequenz der ersten Stufe (f_{IF1}) durch die internen funktionalen Blöcke des TDA und den Keramikfilter vorgegeben ist. Die weiteren Faktoren ergeben sich aus dem Aufbau des Empfängers und werden von Infineon bereitgestellt **TDA-UserManual**

2.3.2 XMC4500

Die Hauptsteuerung der Basisstation übernimmt ein Mikrocontroller der Bauart XMC4500, welcher aus der Mikrocontroller-Familie XMC4000 von Infineon stammt. Diese Baureihe stellt energieeffiziente ICs bereit, welche für industrielle Steuerungen und „Sense & Control“ optimiert sind. Der XMC4500 basiert auf einem Kern CortexTM-M4 des britischen Herstellers ARMTM. Daneben arbeitet der Mikrocontroller mit der von Infineon selbst entwickelte Entwicklungsumgebung DAVETM zusammen. Im speziellen Anwendungsfall kommt die Variante des XMC mit 144 Pins und einem Flash-Speicher von 1024 Kilobit zum Einsatz. Der Chip ist dabei in einem Low Profile Quad Flat Package (LQFP)-Gehäuse verbaut. Durch die Wahl dieses Gehäuses konnte die elektrische Verbindung mit der Platine relativ leicht durch Löten erreicht werden. Im Gegensatz zum ebenfalls erhältlichen Ball Grid Array (BGA)-Gehäuse des XMC sind in diesem alle Kontakte direkt erreichbar und können leicht verlötet werden.

Bei der Auswahl von Pins des XMC zur Verbindung mit den Transceivern wurde vor allem auf die Auswahl für die PP2 Pins geachtet. Um eine spätere Interruptsteuerung möglich zu machen wurden hierfür nur solche Eingänge des Mikrocontroller gewählt, die im Datenblatt mit der Möglichkeit zum Erkennen von Interrupts gekennzeichnet waren.

Zur Verteilung der entstehenden Abwärme wurde auch in diesem Bereich der Platine frei gebliebene Abschnitte zwischen den Leiterbahnen mit geerdeten Kupferflächen gefüllt. Durch teilweise auch mehrfache Durchkontaktierungen wurde sowohl eine saubere Kontaktierung der Flächen durchgeführt, um Flächen schwimmenden Potentials zu vermeiden. Durch die Vias wurde aber auch die thermische Leitfähigkeit zwischen den beiden Kupferlagen erhöht und somit die Abgabe entstehender Wärme von den Bauteilen verbessert.

Beim verwendeten LQFP-Gehäuse des XMC4500 liegt die Rückseite des Halbleiters offen und ist nicht im Gehäuse verschlossen. Im Bereich unter der offenliegenden Rückseite des Chips ist deswegen zur Wärmeableitung ein Feld von 6x6 Vias vorgesehen. Dieser Aufbau dient dazu die Temperatur des Chips (junction temperature) auf den maximal erlaubten Wert $T_j = 150^\circ\text{C}$ zu beschränken.

Um die Ausgabe von aktuellen Systemzuständen zu ermöglichen wurden sieben Status-LEDs an freien Ausgänge des XMC4500 angeschlossen. Diese ermöglichten in active-low Ansteuerung eine Anzeige verschiedener im Mikrocontroller ablaufender Prozesse. Für vier der verwendeten Leuchtdioden wurde grün als Farbe gewählt, für die drei weiteren rot. Zur Vereinfachung eines Resets der Hardware wurde ein entsprechender Taster vorgesehen, mit dem der entsprechenden **PORST**-Pin des Mikrocontroller auf das 0V Potential gezogen wird und somit die Hardware zurückgesetzt wird. Die Programmierung des Mikrocontrollers erfolgt über das Joint Test Action Group (JTAG)-Interface über welches auch das debuggen möglich ist. Der XMC4500 stellt dafür ein JTAG-Modul bereit, welches mit den in IEEE 1149.1 festgelegten Standarts übereinstimmt. Verwendet wird hierfür die achtpolige Variante des Debug-Steckers, bei dem der Platine vom JTAG-Adapter Versorgungsspannung und Masse sowie Signale für Reset, Systemtakt und die Steuerleitung übergeben wird.

Für die Kommunikation des Mikrocontroller mit einem Computer wird die im XMC bereitgestellte Peripherie genutzt. Zur Verbindung mit einem anderen Gerät wurde deshalb eine kombinierte Micro-USB-Buchse verwendet welche sowohl für Typ A oder Typ B Stecker geeignet ist. Um sowohl den Mikrocontroller als auch einen an die Basisstation angeschlossenen Computer gegen Fehlerströme über die USB-Leitung zu schützen wurden die Datenleitungen mit so genannten Transient Voltage Suppressor (TVS)-Dioden, welche gegen Masse geklemmt sind, geschützt. Sowohl positive als auch negative Spannungsspitzen werden dadurch gegen Masse kurzgeschlossen, was zum Schutz des XMC bzw. des angeschlossenen Computer dient. Um eine Verpolung bei Stromversorgung über die USB-Buchse, und somit eine Zerstörung, zu vermeiden wurde eine Schottky-Diode im Strompfad zum Spannungsregler vorgesehen. Diese soll einen Stromfluss im Verpolungsfall unterbinden. Da der Mikrocontroller für die Kommunikation über das USB-Interface die aktuelle Busspannung auf der USB-Leitung benötigt, muss der extra dafür vorgesehene Pin des XMC direkt und ohne schützende Schottky-Diode mit der 5V Leitung der USB-Buchse verbunden werden.

2.3.3 Ethernet

Die Ethernetschnittstelle der Basisstation basiert auf dem Relax Kit von Infineon. Genau wie im Evaluations Board des Herstellers Infineon wurde der Ethernet-Controller KSZ8031RNL von Mircel Inc. verwendet. Dieser stellt alle wichtigen Peripherien selbst zur Verfügung und muss somit nur noch durch ein Quarz und diverse Kapazitäten und Induktivitäten an den Versorgungsleitungen ergänzt werden. Da die im Controller verbaute Stufe zur Interruptgenerierung nur über einen schwachen Pull-Up Widerstand verfügt, musste ein externer Widerstand von $1k\Omega$ verbaut werden. Am Reset-Eingang wurde ebenfalls ein Pull-Up Widerstand verbaut. Dieser wurde um zwei Dioden sowie einen Kondensator zu der im Datenblatt empfohlenen Verschaltung erweitert. So kann

sichergestellt werden, das sowohl beim Anlegen einer Spannung an das Gesamtsystem als auch bei einem Reset des Ethernetbausteins durch den steuernden Mikrocontroller alle Spannungen im sicheren Bereich liegen und die Funktion gewährleistet ist. Die dreizehn zum XMC4500 notwendigen Verbindungen wurden zur besseren Übersicht im Schaltplan in einem Signal-Kabelbaum zusammengefasst. Wegen der Gefahr von Rissen in Lötstellen durch die Platinenbelastung beim Ein- und Ausstecken wurde im Bereich um den Netzwerkstecker die Anordnung von Bauteilen vermieden. Da der KSZ8031RNL nicht lieferbar war und die anfallende Datenmenge nur von geringem Umfang ist, wurde der Controller und die entsprechende Netzwerkbuchse von Würth Electronics zunächst nicht bestückt. Somit wurde eine Verwendung des Ethernet-Controllers auch in der Software des XMC-Mikrocontroller nicht umgesetzt. Da jedoch ein entsprechendes Softwareprojekt für das Relax Kit von Infineon zur Verfügung gestellt wird, wäre eine Netzwerkkommunikation vermutlich mit wenigen Anpassungen leicht umzusetzen.

2.3.4 Spannungsversorgung

Die Bereitstellung der notwendigen Spannung sollte wahlweise über den zur Datenerfassung angeschlossenen Computer oder über ein externes Netzteil erfolgen. Zum Anschluss eines externen Netzteils wurden Lötanschlüsse für eine Steckerleiste im Rastermaß 2,54mm vorgesehen. Genau wie bei der Stromversorgung über die USB-Buchse wurde auch hier eine Schottky-Diode zum Verpolungsschutz der Schaltung integriert. Ausgelegt ist die Basisstation für ein Gleichspannungsnetzteil mit 5V Ausgangsspannung, durch den Aufbau mit den beiden verwendeten Schottky-Dioden und die mögliche Eingangsspannung des nachfolgenden Reglers wäre jedoch auch eine angeschlossene 6V-Stromquelle (bei vernachlässigtem Spannungsabfall an der Diode) unproblematisch. Wegen der bereits erwähnten notwendigen Versorgungsspannung von 3,3V für den XMC4500 und die Transceiver wurde diese mit einem Low Drop-Out (LDO)-Regler angepasst. Dieser verwendete Spannungsregler der Bauart MCP1826S von Microchip Technology Inc. sollte die Eingangsspannung auf das gewünschte Niveau herunter regeln. Als Gehäusetyp wurde das dreibeinige Small Outline Transistor (SOT)-Package gewählt, da keine Variante des LDO mit einstellbarer Ausgangsspannung und somit keine Variante des ICs mit mehr Anschlusspins benötigt wurde. Statt des LDO von Microchip war zunächst ein gleichwertiger Spannungsregler von Infineon, der IFX1117MEV33, vorgesehen. Die beiden LDOs unterscheiden sich in der elektrischen Belegung der Kühlfahne des SOT-223: beim Spannungsregler von Infineon ist diese mit dem 3,3V Output kontaktiert, beim verwendeten LDO mit Ground. Da es sich bei dem Spannungsregler um ein SMD-Bauteil handelt ist die Verwendung von Kühlkörpern schwer möglich. Die Abführung der im Spannungsregler erzeugten Verlustwärme erfolgt deshalb üblicherweise über das Anlöten der Kühlfahne an eine genügend große Kupferfläche, die als Wärmesenke dient. So wird die erzeugte Wärme gespreizt und kann gut an die Umgebung abgegeben werden. Bei Verwendung des zuerst eingeplanten IFX1117MEV33 wäre somit eine Kupferfläche auf 3,3V Potential zum Kühlen notwendig, welche elektrisch isoliert sein müsste. Wegen des XMC4500 und den anderen Bauteilen wäre somit nur eine Kupferfläche mit Abmessungen von etwa 15mm auf 16mm möglich, da die elektrischen Verbindungen bestehender Bauteile des Bereiches nicht unterbrochen werden sollten. Durch die Verwendung des entsprechenden Bauteils von Microchip konnte

auf eine abgetrennte Kupferinsel verzichtet werden und somit die bereits erwähnte GND-Kupferfläche um den Mikrocontroller als gemeinsame Masse- und Kühlfäche verwendet werden. Wegen der vorderseitigen Bauteilbestückung war die verfügbare Kupferfläche auf der Platinenrückseite größer. Um dies beim Ableiten der Wärme vom Bauteil zu nutzen wurden vor allem im Bereich um die Kühlfinne des SOT-223 Gehäuses Durchkontaktierungen angebracht. Diese parallelen „Thermal Vias“ konnten als Wärmepfad zur unteren Kupferfläche dienen. Außerdem wurde der JTAG-Stecker des XMC absichtlich im Bereich neben dem Spannungsregler angebracht. Durch die große Oberfläche stellt auch dieser eine gute Wärmesenke dar.

2.4 Generierte Dokumente

Altium Designer kann aus den erstellten PCB-Daten die für die weitere Verarbeitung benötigten Dateien generieren. Im dazu vorgesehene Output-Job-Manager können entsprechende Outputs gewählt werden und einem Output-Container zugeordnet werden. Dies ist vor allem zum Erstellen gewünschter Dateistrukturen bei größeren Projekten notwendig um diese übersichtlich zu halten. In der vorliegenden Arbeit wurde dies jedoch nur bedingt benötigt. Für die Bestückung der Basisstation wurde zunächst eine „Bill of materials“ also eine Materialliste exportiert mit deren Hilfe die entsprechenden Bestellnummern des Lieferanten Digi-Key herausgesucht und sortiert werden konnten. Mithilfe der ebenfalls generierten „Assembly Drawings“ war die Ausgabe aller Bauteile und deren Platzierung auf der Platine möglich. Ebenfalls wurden in diesem Menü die strukturierten Schaltpläne ausgegeben. Zur dreidimensionalen Visualisierung wurde die fertige Platine mit den in den Footprints enthaltenen Bauteilabmessungen und Höhen als 3D-Modell im Standard for the Exchange of Product Model Data (STEP)-Format ausgegeben. Mithilfe des CAD-Programms „SolidWorks“ wurde daraus ein Gehäuse mit Deckel für die Basisstation erstellt. Diese wurde anschließend mit einem 3D-Drucker ausgedruckt und dient dazu die Platine zu schützen. Die Fertigung der Platine durch den Hersteller erfolgte durch erstellte Gerber Daten, die ebenfalls aus dem Output-Job-Manager generiert werden. Dabei wird für jedes Layer des PCB-Editors eine Gerberdatei erstellt in der die Geometrie der entsprechenden Lage angegeben ist. Jede Lage entspricht in der Herstellung dabei einem Fertigungsschritt. Um Bohrungen in der Platine zu setzen werden zusätzliche „NC Drill-Files“ also Daten für die Numerical Control (NC) der automatischen Maschinen zum setzen von Bohrungen. Diese enthalten den Bohrdurchmesser, die Art der Bohrung sowie den Ort auf der Platine und müssen zusätzlich aus Altium Designer exportiert werden.

2.5 Bestückung

Die gefertigte Platine wurde vor der Programmierung mit den notwendigen Bauteilen bestückt. Dabei wurde zuerst der XMC4500 mit einer Bestückungsmaschine auf dem vorgesehenen Footprint der Platine verlötet. Im folgenden wurden alle weiteren Bauteile sowie die sechs Transceiver mit Lötpaste auf der Platine befestigt und anschließend durch Erhitzen der Platine auf der Heizplatte verlötet. Das Kontaktieren durch die Heizplatte

wurde in mehreren Schritten durchgeführt. Dabei wurden zunächst Bauteile einer guten Toleranz hoher Temperaturen wie Stecker und Widerstände bestückt. Temperaturanfällige Bauteile wie Leuchtdioden und integrierte Schaltkreise wurden nach Möglichkeit zu späterem Zeitpunkt befestigt. Wie bereits erwähnt wurde der Ethernet-Controller nicht bestückt.

Software

3.1 DAVE Entwicklungsumgebung

Das Programm DAVE™ (Digital Application Virtual Engineer) wird von Infineon Technologies AG entwickelt. Es basiert auf der Entwicklungsumgebung oder integrated development environment (IDE) „eclipse“ die von der Eclipse Foundation entwickelt wird. Eine IDE beschreibt dabei allgemein ein Programm zur Softwareentwicklung, welches die einzelnen dazu notwendigen Tools gesammelt zur Verfügung stellt. Dies sind vor allem der Compiler, der Linker, und der Debugger auf die im folgenden noch eingegangen werden soll. DAVE™ stellt eine Möglichkeit zum editieren von Quelltexten und anordnen der einzelnen Programmdateien bereit. Über den enthaltenen GNU C-Compiler wird der erstellte Quellcode in vom XMC lesbare Maschinensprache übersetzt und anschließend durch den Linker zu einem ausführbaren Programm vereint. Durch den enthaltenen Debugger kann das Programm in den Speicher des XMC geladen werden. Dort kann der Programmablauf gestartet und gestoppt werden, außerdem können Werte einzelner Register und Variablen ausgelesen werden.

DAVE™ greift bei der Programmierung von Mikrocontrollern der XMC-Serie auf die so genannten XMC Libraries zurück die von Infineon ebenfalls zur Verfügung gestellt werden. Auf diese soll ebenfalls im weiteren Verlauf eingegangen werden. Ein weiteres Feature in der IDE sind die sogenannten DAVE™ APPs. Mit diesen soll die Programmierung des Mikrocontrollers durch ein Graphical User Interface (GUI) ermöglicht werden. Dazu werden für mögliche von der Hardware zu verrichtende Teilaufgaben APPs von Infineon bereitgestellt. Durch das Einfügen der entsprechenden APPs in das Projekt können diese angepasst und miteinander grafisch verschalten werden. So wird der spätere Programmablauf im Mikrocontroller und dessen Aufgaben festgelegt. Nachdem vom Programmierer noch die Pins ebenfalls grafisch den Aufgaben zugeordnet werden, generiert DAVE™ den Programmcode mit den in den Apps enthaltenen Informationen

Mithilfe des DAVE™ Software development kit (SDK) können nicht nur die Parameter der APPs beim Programmieren, sondern auch diese selbst grundlegend angepasst werden und das entwickeln eigener APPs ist möglich. Im Verlauf dieser Arbeit wurden DAVE™ APPs jedoch nur in einem bereits existierenden Softwareprojekt für ein Relax Kit genutzt,

mit welchem Signale zum Testen der Empfänger an die Basisstation gesendet wurden. In der Basisstation selbst wurden die APPs jedoch nicht benutzt.

3.2 verwendete Peripherie des XMC4500

Der XMC4500-Baustein enthält diverse funktionelle Blöcke die mit einer Bus Matrix an die ARM Cortex M4 Central Processing Unit (CPU) angebunden sind. Dieser Aufbau soll den Prozessor entlasten und im Programmablauf Ressourcen freihalten für andere Operationen. Für die Basisstation waren vor allem der USIC, der Universal Serial Bus (USB) sowie der Ethernet MAC (ETH) die bedeutende Peripherie. Von besonderer Bedeutung sind jedoch die General Purpose Input/Outputs (GPIOs).

3.2.1 GPIO

Die GPIOs werden im so genannten PORTS-Modul der XMC-Architektur gesteuert. In dieser lassen sich die Treiberstufen für die entsprechenden Pins des Mikrocontroller regeln. Dieses Modul ist über die Peripheriebrücke PBA1 ebenfalls an die Bus Matrix und somit den Cortex M4 Kern angebunden. Das Modul stellt für jeden Pin die erste Funktionsauswahl bereit und legt so die Datenrichtung fest. Im „Port Input/Output Control Register“ des Moduls wird für jeden Pin festgelegt ob er als Eingang oder Ausgang verwendet wird. Das momentane elektrische Potential am Eingang wird bei letzterem mit einem Schmitt-Trigger in ein binäres logisches Signal übersetzt. Ist der Pin ein Eingang, so kann dieser dort zusätzlich invertiert werden. Wird ein Pin des XMC als Ausgang konfiguriert, so kann gesteuert werden, ob es sich um einen GPIO-Pin handelt, dessen Status von der Software direkt festgelegt wird. Dabei kann ausgewählt werden ob das logische Ausgangssignal durch einen Treiber in Open-Drain Konfiguration oder durch Push-Pull erzeugt werden soll.

Zur Nutzung eines Ausganges durch die im Mikrocontroller verfügbaren Peripherie sind diese direkt mit den entsprechenden Modulen verbunden. Dadurch kann das Modul selbst den elektrischen Zustand am Eingang auslesen und verwerten oder festlegen wenn es sich um einen Ausgang handelt. **XMC-Reference** Auch das weitere Verhalten von Pins, wie etwa beim Anschalten, bevor die Versorgungsspannung ein gültiges Level erreicht hat, lassen sich im PORTS-Modul anpassen.

3.2.2 USIC

Die ICs der XMC-Familie verfügen über ein Modul zu Kommunikation über diverse serielle Protokolle, den Universal Serial Interface Channel (USIC). Dieses ist programmierbar und erlaubt damit eine individuelle Verwendung, kann aber gleichzeitig die notwendigen Arbeiten für den Prozessor übernehmen. Der XMC4500 verfügt über insgesamt sechs USIC-Kanäle und kann somit mehrere Protokolle gleichzeitig verwenden. Die Mikrocontroller unterstützen die Protokolle UART, I²C, IIS, LIN und das für die Basisstation verwendete SPI in einfach, doppelt und quad-Ausführung. Für diese Arbeit wurde alle Kommunikation mit einem gemeinsamen USIC-Kanal umgesetzt. Da die einzelnen USICs und deren Kanäle verschieden viele Slave-Select-Leitungen besitzen wurde der Kanal 0 des USIC 0

ausgewählt, nur dieser verfügt über die benötigten sechs Select-Leitungen. Eine Umsetzung mit einem anderen USIC wäre ebenfalls möglich gewesen. Die Wahl des Transceivers hätte dann aber manuell erfolgen müssen und hätten nicht vom das Modul geregelt werden können.

3.2.3 ERU

Von zentraler Bedeutung für die Funktion der Basisstation war die Behandlung von Interrupts durch den IC. Der XMC4500 besitzt dafür zwei entsprechende Event Request Unit (ERU)-Module die eine solchen erkennen können und den Prozessor zum Aufrufen einer Interrupt Service Routine (ISR) auffordern können. Jedes Modul verfügt über vier Kanäle auf denen bei einem Interrupt ein vierstufiger Prozess durchlaufen wird: In der ersten Stufe der ERU, der so genannten Event Request Select (ERS) lassen sich aus zwei Eingänge mit jeweils vier Signalen die gewünschten Eingänge wählen. In der Event Trigger Logic (ETL) generiert der IC aus dem Signalstatus ein Trigger-Event indem Veränderungen erkannt werden. So kann eine fallende oder steigende Flanke, die einen Interrupt auslösen soll, erkannt werden. In der Cross Connection Matrix können Signale der verschiedenen ETLs zu den vier Output Gating Units (OGUs) weitergeleitet und somit dort untereinander und mit Triggersignalen von anderen Peripherie-Modulen des XMC kombiniert werden. In der OGU wird durch Vergleich der verschiedenen aufgetretenen Trigger und Muster entschieden ob ein kompletter Interrupt aufgetreten ist und leitet diesen weiter oder ob etwa nur das gewählte Muster erkannt wurde, was für andere Module wichtig ist. Diese Informationen werden entsprechend an die Peripherie weitergeleitet, sind aber für die Funktion der Basisstation nicht weiter von Bedeutung. Bei Vorliegen aller Bedingungen für einen Interrupt wird diese Information an den Nested Vectored Interrupt Controller (NVIC) im Cortex-M4 weitergeleitet. Dieser Teil des Prozessorkerns erkennt die Interruptaufforderung und sorgt dafür, das der aktuelle Prozessorstatus gespeichert wird. Nach Ablauf der ISR wird der Prozessorstatus wieder hergestellt.

3.2.4 USB

Das USB-Modul des XMC4500 arbeitet nach den Spezifikationen für USB 2.0 und den „On-The-Go“-Spezifikationen der Version 1.3. Der Mikrocontroller könnte durch das USB-Modul sowohl als Host oder als USB-Slave arbeiten. In diesem Fall wurde der IC als Slave betrieben. Das USB-Modul besitzt über eine eigene Interruptsteuerung und ist damit eine der oben erwähnten Peripherien des XMC deren Steuerung auch über Interrupts gelöst ist. Die gesamte Übertragung wird durch den USB-Kern gesteuert und empfangene oder zu sendende Pakete werden in einem FIFO-Puffer gespeichert. Für die Kommunikation der Basisstation mit dem Host-Computer wird ein virtueller COM-Port durch das USB-Interface emuliert.

3.2.5 Ethernet

Im XMC4500 werden Netzwerkverbindungen durch das Ethernet-Modul behandelt. Dieses unterstützt Datenübertragungen mit Geräten über IPv4 und IPv6 sowie Übertragungsraten von 10/100 MBit/s. Dazu werden zunächst die Daten von der CPU über ein

Bus-Interface übertragen. Im „MAC Transaction Layer“ werden die vom Prozessor bzw. über Ethernet empfangenen Datenpakete zwischengespeichert. Der Ethernet-Kern formatiert die zu sendenden Daten und stellt sie einem „Physical Layer“ zu, welches die Daten für den Kanal moduliert.

3.3 verwendete Bibliotheken

3.3.1 XMC Library (XMC Lib)

Infineon stellt für seine ICs der XMC4000 Serie zu der auch der XMC4500 gehört die „XMC Peripheral Library“ bereit. Diese erlaubt einen vereinfachten Zugriff auf alle Module und die entsprechenden Register und soll dadurch den Modulzugang übersichtlich gestalten und den Programmcode vereinfachen und leichter lesbar machen. Die Software baut auf dem Cortex Microcontroller Software Interface Standard (CMSIS) auf, erlaubt die Verwendung verschiedener Compiler und kann mit oder ohne DAVETM bzw. mit oder ohne DAVE APPs verwendet werden. Für die Programmierung wurde mit der Software „doxygen“ eine Dokumentation zur Bibliothek als HTML generiert.

3.3.2 SPI Library

Die verwendete Bibliothek zur Steuerung des SPI-Interfaces basiert auf dem SPI-Modul der XMC Library. Die von Infineons XMC Library zur SPI-Kommunikation zur Verfügung gestellten Funktionen steuern das USIC-Modul des Mikrocontrollers an. Die Bibliothek zur SPI-Kommunikation muss somit nur noch die Funktion

```
1 XMC_SPI_CH_Transmit(channel, data, XMC_SPI_CH_MODE_STANDARD);
```

der XMC Library aufrufen um eine Übertragung über das Serielle Interface durchzuführen. Daneben liegt die Hauptaufgabe der Bibliothek vor allem in der Auswahl des entsprechenden Transceivers über das Slave-Select-Signal. Dazu initialisiert die Bibliothek zuerst den USIC entsprechend den in der Headerdatei vorgegebenen Pins für MISO, MOSI und den Pins zur Auswahl des jeweiligen Slaves.

SPI Übertragung

Die durch die Software umgesetzte serielle Übertragung zwischen den ICs entspricht dem SPI-Protokoll das ursprünglich von Motorola entwickelt wurde. **BuchSPI** Durch das Chip Select-Signal wählt der XMC den entsprechenden Transceiver aus. Da der TDA5340 active-Low arbeitet, also bei anliegendem Masse (GND)-Potential als ausgewählt gilt, wird dieser Eingang im Datenblatt auch als Non-Chip-Select (NCS) bezeichnet. Der USIC des XMC beginnt nun die Clockleitung des Busses zu treiben. Eine Periode dieser zyklischen Rechteckspannung begrenzen dabei die Zeit in der das auf den Master-Out Slave-In (MOSI) und Master-In Slave-Out (MISO)-Leitungen ein Bit übertragen wird. Dabei stellt nach der Definition aus dem Datenblatt des Transceivers ein 0V-Spannungspegel eine logische 0 dar. In der Kommunikation mit den TDA5340 werden jeweils acht Bit zu einem Datenwort zusammengefasst. Der Transceiver erwartet mit dem ersten Datenwort eine Instruktion und je nach Art dieser noch einen Parameter durch die Übertragung

des folgenden Wortes. Mögliche Anweisungen sind dabei das Lesen oder Schreiben eines Registers dessen Adresse als Parameter übertragen wird, das Auslesen oder Beschreiben der FIFO-Speicher oder das Setzen des TDA5340 in den so genannten „Transparent Mode“ auf den nicht weiter eingegangen werden soll. Nach Übertragung von Instruktion und Parameter werden entweder vom Master oder vom Slave, also vom Transceiver, die gewünschten Daten übertragen.

Beim Schreibzugriff auf den TDA kann durch geeignete Instruktion gewählt werden ob nur auf ein Register geschrieben wird oder ob der Transceiver nachfolgende Datenwörter als Werte in die folgenden Register übernehmen soll. Beim Lesezugriff besteht die selbe Auswahlmöglichkeit. Bild zeigt den Start einer solchen SPI-Kommunikation zwischen dem Transceiver und dem Mikrocontroller. Die Dauer der Übertragung betrug etwa 0,26ms.

3.3.3 TDA5340 Library

Die Hauptaufgabe in der Kommunikation mit den Transceiver-ICs wurde durch die vorgegebene Bibliothek für den TDA5340 übernommen. Diese stellte Funktionen zum Senden und Empfangen von Daten mit dem Transceiver zur Verfügung. Die Bibliothek liest dazu den Empfangs-FIFO aus oder schreibt in den Puffer für zu sendende Daten. Auch der Lese- und Schreib-Zugriff auf die Steuerregister der Transceiver kann über die Bibliothek geregelt werden. Dazu stellt die Bibliothek auch entsprechende Makros bereit, welche die Namen der Register in die hexadezimale entsprechende Adresse umwandeln, was die Lesbarkeit erheblich beeinflusst. Daneben wurde über die Bibliothek auch die notwendigen Einstellungen für das Erkennen von Interrupt im XMC4500 vorgenommen. In diesem Bereich waren die notwendigen Anpassungen der Library am tiefgreifendsten, da diese nur für die Interruptbehandlung mit einem Transceiver ausgelegt war. Bei anderen Funktionen der Bibliothek waren nur kleinere Anpassungen notwendig, sodass etwa sichergestellt wurde, für welchen Transceiver die aufgerufene Funktion ausgeführt werden sollte, etwa bei welchem das entsprechende Register ausgelesen wurde. Zur Verbindung mit den TDA5340 basierte die Bibliothek auf der SPI Library. Dieser wurde die Nummer des Transceivers als Chip-Select übergeben um sicherzustellen, dass mit dem richtigen kommuniziert wurde.

3.3.4 Virtueller COM Port

Die Kommunikation der Basistation mit dem Hostcomputer zum Übertragen der gemessenen Werte wurde nach dem Vorbild eines Beispielprojektes für DAVE™ umgesetzt. Die Bereitstellung des virtuellen seriellen Ports erfolgt auf Seiten des XMC über die LUFA (Lightweight USB Framework for AVR)-Bibliothek. Mit dieser beschränkt sich die Ausgabe über den COM-Port auf das Übergeben der zu sendenden Zeichen an eine entsprechende Funktion. Eine Steuerung des XMC durch Empfangen von Daten über den COM-Port wäre mit der Bibliothek ebenfalls möglich, war jedoch nicht notwendig. Die Bibliothek und das Beispielprojekt wurde dahingehend angepasst, dass ganze Zeichenketten statt nur einzelner Zeichen der Funktion zum Senden übergeben werden konnten. Auch wurde das Senden von Integer-Variablen ermöglicht, indem diese zu Zeichen umgewandelt

wurden.

3.4 Programmablauf

Im Programmablauf des XMC wurde zunächst eine Warteschleife umgesetzt, in welcher der Mikrocontroller auf eine erste Eingabe durch den Benutzer am Hostcomputer wartet. Bereits hier wurde die Übertragung über den seriellen COM-Port initialisiert. Nach dem diese Bestätigung über den COM-Port vom Mikrocontroller empfangen wurde setzt dieser zunächst alle Transceiver in den Schlafmodus von welchem aus eine Kommunikation möglich ist. Daraufhin beginnt der XMC4500 mit der Konfiguration der für die Interrupts notwendigen Pins und ermöglicht somit das Empfangen von Interrupt Requests (IRQs) durch die TDA5430. Anschließend initialisiert er zunächst das SPI-Modul um im folgenden die Transceiver darüber konfigurieren zu können. Bevor das dazu notwendige Schreiben in die Register der TDAs jedoch möglich ist, wird eine gewisse Verzögerung benötigt. Diese resultiert daher, dass der Wechsel des TDA vom ausgeschalteten Zustand in den Schlafmodus eine gewisse Zeitspanne benötigt. Im folgenden werden alle Transceiver initialisiert und anschließend in den Receiver-Modus gefahren. Daraufhin setzt der IC noch alle für die Übertragung notwendigen Variablen und alle Felder zum Speichern von Daten zu null. Nun Beginnt der Prozessor mit einer Dauerschleife in der dieser auf das Ankommen von Übertragungen wartet und teilt dies auch dem Benutzer am Hostcomputer über eine Ausgabe mit. In dieser Endlosschleife wechselt sich die interruptbasierte Datenerfassung mit der Weiterleitung der erfassten Daten zyklisch ab. Wobei nur bei erfolgreichem Empfang ein Senden an den steuernden Computer erfolgt.

Bei der Programmierung wurde darauf geachtet möglichst alle Konstanten wie Adressen von Pins und Ports durch Makros für den Präprozessor auszulagern. Dadurch konnte einerseits die Lesbarkeit des Quellcodes erhöht werden, andererseits sind so aber Änderungen durch vertauschen von Pins, sowohl in der Hardware als auch nur in Software, leichter möglich. Da es sich beim XMC4500 um einen Mikrocontroller mit einer 32Bit Architektur handelt wurden Integervariablen ausschließlich vom Typ *uint32_t* oder *int32_t* verwendet. So konnte eine genaue Anpassung der Software an die Registergrößen im XMC vorgenommen werden.

3.4.1 Konfiguration der Funkmodule

Bei der Initialisierung erhalten diese die gewünschten Werte für die Sende- und Empfangsfrequenzen. Diese werden über die Teilerrate für die Phasenregelschleife (PLL) übergeben und ermittelt. Über die entsprechenden Register wird auch das Verhalten bei

3.4.2 interruptbasierte Datenerfassung

Die Erfassung der Daten erfolgt im Programmablauf innerhalb der dauerhaften Ablaufschleife. In der Interrupt Service Routine (ISR), in welche der Prozessor beim Auftreten eines Interrupts springt, wird lediglich einer globalen Variable der Wert 1 zugeordnet. Für jeden der möglichen Transceiver existiert eine solche Flagge, die einen aufgetretenen Interrupt anzeigt. Nach dem Setzen in der ISR kehrt der Prozessor zum Ablauf in die

Dauerschleife zurück. Innerhalb dieser wird nun zyklisch abgefragt ob diese Flagge gesetzt wurde. Beim Auftreten einer solchen Flagge, also nach dem aufgetretenen Interrupt, liest der XMC4500 die Interrupt Status Register des entsprechenden Transceivers aus. Da davon auszugehen ist, dass alle sechs Transceiver-ICs zeitgleich eine Übertragung erhalten wurde, wurde diese mehrstufige Abfrage gewählt. So wird zuerst nur in der ISR die Flagge gesetzt, um die dadurch verstreichende Zeit möglichst kurz zu halten und zu ermöglichen, dass eine solche Flagge auch jederzeit im Programmablauf gesetzt werden kann. In der zweiten Stufe liest der Mikrocontroller nun die drei Interrupt Status Register aus. Dies ist notwendig, da es sich dabei um so genannte „Read-Clear“-Register handelt, welche nach dem Auslesen über SPI automatisch zurückgesetzt werden. In den Interrupt-Registern sind die Ereignisse kodiert, die einen Interrupt ausgelöst haben. Bei der Konfiguration der Funkmodule wurde eingestellt, bei welchen Ereignissen der Transceiver einen Interrupt auslöst. Typische Ereignisse sind ein fast gefüllter Empfangs-FIFO, was eine angekommene Übertragung anzeigt, oder auch ein leerer FIFO, was anzeigt, dass dieser ausgelesen wurde. Die dritte Stufe der Datenerfassung ist nun die Abfrage der Daten vom Transceiver. Da diese SPI-Datenübertragung deutlich mehr Zeit in Anspruch nimmt, muss diese getrennt vom Erkennen der Interrupts und dem Auslesen der Interrupt-Register erfolgen. Da die gewünschten Werte in den Registern gespeichert sind, ist das Auslesen zeitkritisch, da eine erneute Übertragung diese überschreiben würde. Jedoch ist das Auslesen der Werte bei weitem nicht so zeitkritisch wie die ankommenden Interrupts, da diese im Verlauf einer Übertragung auch mehrfach ankommen können. Das Abfragen der empfangenen Daten aus dem FIFO-Speicher hat eine noch geringere Priorität, da dieser die Datenpakete bis zum Auslesen behält. Eine Abfrage der Messwerte und ein Auslesen des FIFO wird bei abgeschlossener Übertragung durchgeführt, diesen Zeitpunkt erkennt der Mikrocontroller durch das entsprechend gesetzte Bit des Interrupt-Registers. Der XMC4500 speichert alle abgefragten Werte wie Feldstärke, die Einstellungen der automatic gain control (AGC) und die angekommenen Daten in den vorbereiteten dedizierten Speichern. Abschließend werden die Transceiver wieder in den Empfangsmodus gesetzt.

3.4.3 Weiterleitung der erfassten Daten

Da eine Übertragung über den COM-Port bereits weit vor der Dauerschleife vom Mikrocontroller gestartet wurde, diente dazu dem Benutzer die Bereitschaft zum Empfangen mitzuteilen, sobald sämtliche Initialisierungen abgeschlossen waren. Somit musste dieser auch nun nicht mehr selbst initialisiert werden. Im zweiten Teil der Ablaufschleife des Programmablaufs wurde nun die Weitergabe der empfangenen Daten und der gemessenen Werte behandelt. Dazu wurde nach dem Abholen der empfangenen Daten von den Transceivern eine Statusflagge in Form einer Integer-Variable gesetzt. Nur beim Auftreten dieser Flagge wurde der Programmteil zum Senden über den COM-Port ausgeführt. Dort wurde nun jeweils abwechselnd ein Wert des Speichers und ein String mit einer Beschreibung oder dem Namen des Wertes über COM ausgegeben. Ein Ausschnitt des Quellcodes ist im Quellcode 3.1 zu erkennen. Die Zeichenfolge `\r\n` stellt dabei den Übergang in eine neue Zeile in der Ausgabekonzole dar. Wie in C üblich werden Strings in doppelten Anführungszeichen im Quelltext eingefügt. Eine entsprechende Ausgabe ist in Bild ... zu erkennen.


```

1  COM_send_string("##### Übertragung erkannt #####\r\n");
2  COM_send_string("Übertragung Nummer ");
3  COM_send_int_as_string(transfervummer);
4  COM_send_string("\r\n\r\n");
5  COM_send_string("TDA1:");
6  COM_send_string("\r\nPMF:");
7  COM_send_int_as_string(rssiTDA1.pmf);
8  COM_send_string("\r\nPRX:");
9  COM_send_int_as_string(rssiTDA1.prx);
10 COM_send_string("\r\nRX:");
11 COM_send_int_as_string(rssiTDA1.rx);
12 COM_send_string("\r\nPPL:");
13 COM_send_int_as_string(rssiTDA1.ppl);
14 COM_send_string("\r\nAGC:");

```

Quellcode 3.1: Ausschnitt aus dem Senden der Daten über den COM-Port

Am Hostcomputer wurden die Daten mit der Software „PuTTY“ entgegengenommen und dem Benutzer in einer Konsole angezeigt. Dazu wurde der von Infineon zur Verfügung gestellte Treiber verwendet um dem XMC4500 als COM-Port zu erkennen. Die Einstellungen der Seriellen Übertragung wurden vom Beispielprojekt übernommen, sodass am Computer mit einer Baudrate von 115200Bd und acht Daten Bits pro Zeichen empfangen wurde und was im Programm eingestellt werden musste. Nach der bereits erwähnten anfänglichen Bestätigung der Kommunikation durch den Nutzer, wurden bei jeder vom Mikrocontroller erkannten Übertragung die gemessenen Feldstärkewerte ausgegeben. Neben den PMF, PRX, RX, PPL und AGC-Werten wurde noch die Empfangsleistung ausgegeben. Diese wurde zur Laufzeit aus dem PPL und dem AGC-Wert errechnet und in dBm angezeigt. Dazu diente eine vorgegebene Funktion, welche aus dem PPL-Wert und dem Wert der AGC mittels kalibrierter Parameter eine Feldstärke berechnete. Die Werte wurden in einem Vorprojekt durch Messungen kalibriert und sind hier nicht weiter von Bedeutung. Die Abkürzungen der übertragenen Werte sind in Tabelle 3.1 aufgeführt. Die

PMF	Peak Memory Filter
PRX	Peak Detector
RX	Received Signal Strength Indication (RSSI)
PPL	RSSI Payload Peak Detector
AGC	Automatic Gain Control

Tab. 3.1: Verwendete Abkürzungen der übertragenen Messwerte

empfangenen Daten wurden ebenfalls ausgegeben. Alle Werte und Daten wurden nach Transceiver getrennt ausgegeben um eine Vergleichsmöglichkeit zu geben und um die Ausgabe möglichst übersichtlich zu gestalten. Auch wurden die empfangenen Übertragungen durchnummeriert und die entsprechende Übertragungsnummer mit ausgegeben. So ließen sich einerseits die Sendepositionen den gemessenen Werte zuordnen. Andererseits waren so aber auch verlorengegangene Übertragungen sichtbar. Eine typische Ausgabe der Konsole ist im Bild ?? erkennen. Zur Darstellung einer eingegangenen Übertragung an der Basisstation sollte die LED Nummer 7 nach jedem Empfangen kurz rot aufblinken.

Feldtest

4.1 Aufbau

Zur Evaluation der Basisstation wurde ein XMC4500 Relax Kit von Infineon mit einem aufgesteckten Evaluations-Board für den TDA5340 betrieben. Mit Hilfe einer Powerbank konnte dieses mobil über den USB-Anschluss des Relax Kit betrieben werden. Dieses wurde auf eine Sendefrequenz von ... und eine Empfangsfrequenz von ... programmiert was durch die Werte für die PLL im TDA5340 eingestellt wurde. Die Basisstation wurde mit sechs Antennen, die einen Verstärkungsfaktor von 3,6dBi und eine Mittenfrequenz von 868MHz aufwiesen, bestückt. Die Basis wurde über USB an den Computer zur Auswertung angeschlossen. Das Auslesen der durch virtuellen COM-Port übertragenen Daten erfolgte mit PuTTY. Die Messungen fanden innerhalb des Gebäudes statt.

4.2 Durchführung

Es wurden im selben Raum von diversen Positionen durch einen Tastendruck am Relax Kit eine Übertragung ausgelöst. Dabei wurden die zuvor einprogrammierte Zeichenkette 1,2,3,4,5,6,7,8,9 ausgesendet. Der Abstand zur Basisstation betrug im ersten Test 3,30m und wurde nach jeder Übertragung um 30cm verringert. In einem zweiten Test wurde ebenfalls mit einem Abstand von 3,3m gestartet. Nach jeder Übertragung wurde der Sender 30cm von der Startposition aus, entlang einer Linie, rechtwinklig zur Sichtverbindung Startpunkt-Basis, vom Relax Kit entfernt. Die gemessenen Werte wurden zur weiteren Auswertung abgespeichert. In beiden Test war die Basisstation so ausgerichtet, das TDA1 in Richtung der gemeinsamen Startposition der Tests zeigte. Logic Analyser und Debugger waren während der Tests nicht an der Basisstation angeschlossen. So sollten Abschattungseffekte durch dieser verhindert werden. Die Basisstation wurde auf einem 70cm hohen Tisch aufgestellt. Der Sender wurde auf gleicher Höhe freischwebend bewegt. Es wurden in beiden Tests zehn Messpunkte gesendet. Sowohl die Antennen an der Basisstation, als auch jene am Relax Kit waren senkrecht nach oben zeigend ausgerichtet.

4.3 Ergebnisse und Auswertung

Auffallend ist, dass zwar in jedem Test zehn mal durch das Drücken des Tasters eine Übertragung ausgelöst wurde, jedoch öfter eine Übertragung an der Basisstation registriert wurde. Im Ersten Feldtest wurden fünfzehn, im zweiten sogar sechzehn gültige Übertragungen von der Basisstation an den Hostcomputer weitergegeben. Im ersten Test konnten bei der ersten Übertragung an den Transceivern eins bis vier keine Daten empfangen werden. Erst in der darauffolgenden zweiten erkannten Übertragung wurde hier die gesendete Zahlenfolge empfangen. Bei der zweiten Übertragung, welche über die Konsole ausgegeben wurde, stimmten die Werte von TDA5 und TDA6 in allen ausgelesenen Registern mit den Messwerten der ersten Übertragung überein. Daraus ist zu folgern, dass es sich bei diesen Werten noch um die Messungen aus dem ersten Transfer handelt. Somit wäre zu folgern, dass durch eine leichte Verzögerung zwischen den TDA5340 der steuernde Mikrocontroller eine gemeinsame Übermittlung als zwei getrennte Übertragungen interpretiert hat. Transceiver 1 konnte in dem Test erst ab der vierten Übertragung gültige Daten empfangen. Außerdem waren die gemessenen Empfangsleistungen stets geringer als -100dBm , lediglich bei der letzten Übertragung, welche bei einem Abstand von 30cm stattfand, konnte hier ein Wert von -97dBm gemessen werden. Da diese sehr schwachen Empfangsleistungen in vorherigen Tests nicht auftraten ist zu vermuten, dass etwa eine nicht richtig verbundene Antenne Grund des schwachen Empfangswertes war. TDA3 konnte in keiner einzigen Übertragung passable Messwerte liefern. Die errechnete und ausgegebene Empfangsfeldstärke von -114dBm entsprach dem minimal möglichen Ausgabewert der dafür verwendeten Funktion. Es ist also davon auszugehen, dass in diesem Transceiver nie eine Funkverbindung erkannt wurde. Gründe dafür wären ein Fehler im Anpassnetzwerk zwischen der Antenne und dem Transceiver oder ein Defekt des selbigen. Letzteres ist eher unwahrscheinlich, da eine Verbindung über SPI mit dem IC möglich war. Lediglich ein Teildefekt in der RF-Sektion des Chips wäre also denkbar. Daneben waren vereinzelt auch noch Übertragungen zu erkennen, in denen Registerwerte mit den Messungen aus den folgenden oder vorherigen Übertragungen übereinstimmten. Auch hier ist zu vermuten, dass einzelne Transceiver eine Übertragung nicht erkennen konnten.

Zu beachten ist, dass die vermeintlich doppelt ankommenden Übertragungen auch vom Sender ausgehen konnten. Es ist nicht komplett sicher festzustellen ob die mehrfache Ausgabe einer Übertragung durch die Basisstation bedingt ist oder ob vom Relax Kit mehr als die gezählten Übertragungen versendet wurden. Das Versenden der Nachricht wurde an diesem durch einen Tastendruck ausgelöst. Durch ein Prellen des Tasters konnten auch mehrfache nur minimal verzögerte Übertragungen ausgelöst worden sein. Für die Durchführung von nachfolgenden Tests wäre demnach ein Sicherstellen einer nur einfachen Übertragung notwendig. Zusätzlich wäre eine Ausgabe der Sendungsnummer am Relax Kit und ein neuer zu sendender Datensatz für jede Übermittlung hilfreich.

Zusammenfassung und Ausblick

Zusammenfassend kann man sagen, dass die Basisstation den gewünschten Zweck gut erfüllt. Dabei besteht trotzdem noch die Möglichkeit zur weiteren Anpassungen an den aktuellen Nutzen. So können alle Transceiver ordnungsgemäß angesteuert werden und die Erkennung der Interrupts durch den Mikrocontroller funktioniert. Trotz der zum jetzigen Zeitpunkt nicht umgesetzten Netzwerkverbindung der Basisstation konnte eine funktionierende und einfache Ausgabe der Werte umgesetzt werden, welche trotzdem eine frei anpassbare und übersichtliche Ausgabe erlaubt. Problematisch war hierbei die Verwendung eines abgekündigten und nicht mehr hergestellten ICs. Mittlerweile existieren jedoch Nachfolgemodelle des Ethernet-Controllers, sodass eine Migration möglich sein sollte, entsprechende Datenblätter mit Hinweisen stellt der Hersteller bereit.

Ausblickend ließen sich an der Basisstation noch weitere Verbesserungen durchführen. So wäre noch das Eruiere des Grundes für den schlechten Empfang am Transceiver 3 notwendig. Dafür könnte durch das Anschließen eines Signalgenerators an die Antennenbuchse der entsprechenden Transceiverbaugruppe ein möglicher Fehler im Anpassnetzwerk aufgezeigt werden. Sollte auch dies nicht zu einer Verbesserung führen, wäre ein Austausch des ICs notwendig. Für das bessere Erkennen von doppelten Übertragungen wäre das Einfügen eines Zeitstempel in die Ausgabe hilfreich. Dadurch könnten Übertragungen die kurz hintereinander eintreffen markiert und entsprechend zu einer korrekten zusammengefügt werden. Zu diesem Zweck würde es sich anbieten die Realtime-Clock des XMC4500 zu verwenden. Dazu würde das auf der Platine vorsorglich verbaute Uhrenquarz verwendet werden. Zwar wäre auch durch das Abwarten auf weitere verzögerte Übertragungen von anderen Transceivern das Problem der auf mehrere Ausgaben verteilten Übertragungen vermeidbar. Dies würde jedoch zu einer Todzeit führen, in der keine andere Übertragung möglich ist, was zu vermeiden ist. Auch wäre für eine Veränderung an der Platine die Auswahl eines anderen Eingangspins am XMC für das vom PP2 Pin des Transceiver 6 kommenden Interruptsignals sinnvoll. So wären die TDA3 und TDA6 nicht an den selben Interruptkanal des Mikrocontroller angeschlossen. Die dazu notwendigen Änderungen an der Software würden sich auf die Änderungen der entsprechenden Makros in der Headerdatei beschränken. Alternativ ließe sich möglicherweise das Problem der konkurrierenden Interrupts über Anpassungen in der Software lösen. So wäre es möglich das Interruptsignal

einzelner TDA5340 nicht über den PP2 Pin auszugeben, sondern auch über die ebenfalls mit dem XMC verbundenen PP0 und PP1 Pins. Somit wäre eine Verteilung auf einzelne Kanäle der ERU wahrscheinlich möglich.

Abbildungsverzeichnis

6.1	Layout des Aufsteckboards mit dem TDA5340	34
6.2	Layout des Aufsteckboards mit dem TDA5340	35

Tabellenverzeichnis

3.1	Verwendete Abkürzungen der übertragenen Messwerte	22
6.1	Seriennummern der im Projekt verwendeten TDA5340	33

Quellcodeverzeichnis

3.1	Ausschnitt aus dem Senden der Daten über den COM-Port	21
../..	/Dave/Basisstation/Basisstation/ISRs.c	34
../..	/Dave/Basisstation/Basisstation/Init.c	36

Anhang

EINFÜGEN: Layout Stefan Erhard Bilder Altium 3D Modelle Dateien und Dateiverzeichnis (s. XMC Peripheral lib startseite)

6.1 Seriennummern

Alle TDA5340 verfügen über eine eingebaute Seriennummer, welche ausgelesen werden kann. Die Seriennummern der verwendeten TDA5340 sind in der Tabelle aufgeführt.

TDA	Seriennummer
TDA1	33020236
TDA2	11727080
TDA3	11545236
TDA4	11728870
TDA5	11550773
TDA6	33026263

Tab. 6.1: Seriennummern der im Projekt verwendeten TDA5340

6.2 3D-Daten

6.2.1 Platine

6.2.2 Gehäuse

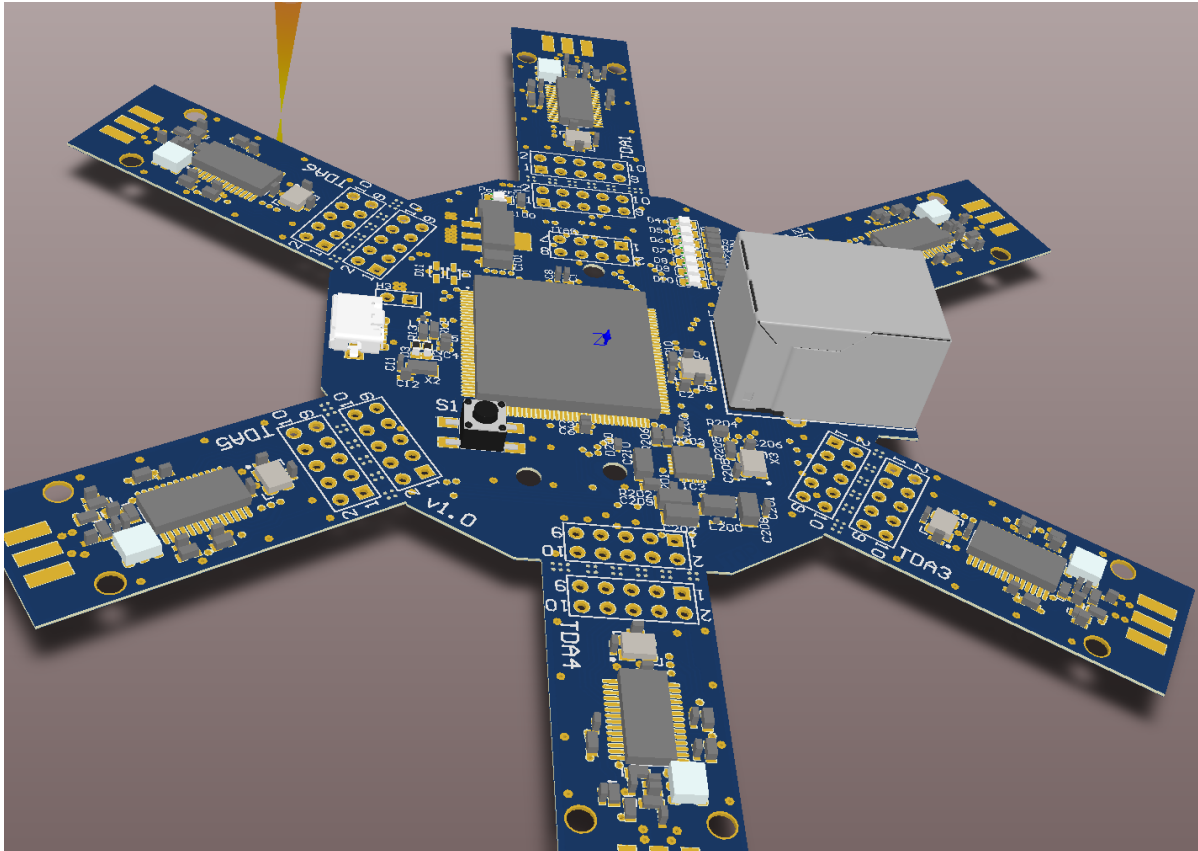


Abb. 6.1: Layout des Aufsteckboards mit dem TDA5340

6.3 Layout TDA5340 Aufsteckboard

Das Layout der Transceiver-Unterbaugruppen orientiert sich an dem Layout eines Aufsteckboards für den „XMC 2Go“ damit ein TDA5340 auf mit diesem verwendet werden kann.

6.4 Quellcode

6.4.1 Main.c

6.4.2 ISRs.c

```

1  /*
2   * ISRs.c
3   *
4   * Created on: Jul 7, 2016
5   * Author: student06
6   */

8  #include "Header_general.h" //including all Header files

10 // ISR für TDA1 (ERU1 OGU0 IRQ)
11 extern void ERU1_0_IRQHandler(void) {

```

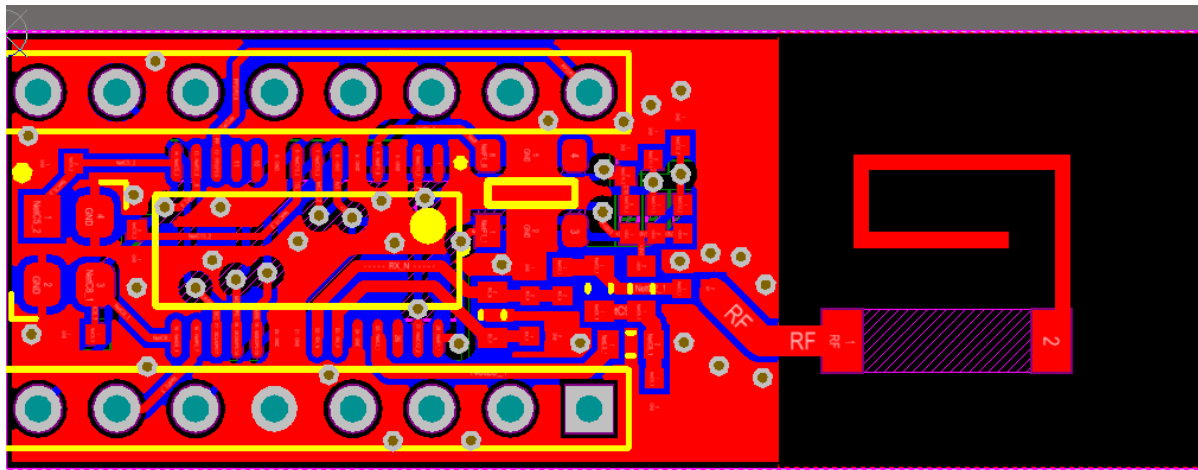


Abb. 6.2: Layout des Aufsteckboards mit dem TDA5340

```

12  query_interruptTDA1_flag = 1;
13  // COM_send_string("INTERRUPT1\r\n");
14  }
15  // ISR für TDA2 (ERU0 OGU0 IRQ)
16  extern void ERU0_0_IRQHandler(void) {
17  query_interruptTDA2_flag = 1;
18  // COM_send_string("INTERRUPT2\r\n");
19  }
20  // ISR für TDA3 + TDA6 (ERU0 OGU1 IRQ)
21  extern void ERU0_1_IRQHandler(void) {
22  //XMC_ERU_ETL_ClearStatusFlag(XMC_ERU0, 1);
23  // COM_send_string("ISR 3 und 6 \r\n");

25  // //Check which Interrupt has occurred
26  // uint32_t status_tda3 = XMC_GPIO_GetInput(PORT_PP2_TDA_3, PIN_PP2_TDA_3);
27  // uint32_t status_tda6 = XMC_GPIO_GetInput(PORT_PP2_TDA_6, PIN_PP2_TDA_6);
28  // if ((!status_tda3) && (status_tda6)) {
29  //   COM_send_string("INTERRUPT3\r\n");
30  //   query_interruptTDA3_flag = 1;
31  // }
32  // if ((status_tda6 == 0) && (status_tda3 != 0)) {
33  //   COM_send_string("INTERRUPT6\r\n");
34  //   query_interruptTDA6_flag = 1;
35  // }
36  // if ((status_tda6 == 0) && (status_tda3 == 0)) {
37  //   COM_send_string("INTERRUPT 3&6 \r\n");
38  //   query_interruptTDA3_flag = 1;
39  //   query_interruptTDA6_flag = 1;
40  // }
41  query_interruptTDA3_flag = 1;
42  query_interruptTDA6_flag = 1; //beide setzen egal welcher ankommt -> es werden
    beide ausgelesen
43  // led_on(LED7);
44  }
45  // ISR für TDA4 (ERU1 OGU1 IRQ)
46  extern void ERU1_1_IRQHandler(void) {
47  query_interruptTDA4_flag = 1;
48  // COM_send_string("INTERRUPT4\r\n");
49  }
50  // ISR für TDA5 (ERU0 OGU2 IRQ)
51  extern void ERU0_2_IRQHandler(void) {
52  // COM_send_string("INTERRUPT5\r\n");
53  query_interruptTDA5_flag = 1;
54  }
55  // ISR für TDA6 (ERU0 OGU3 IRQ)

```

```

56 extern void ERU0_3_IRQHandler(void) {
57     led_on(LED5);
58     query_interruptTDA6_flag = 1;
59 }

```

6.4.3 Init.c

```

1  /*
2   * Init.c
3   *
4   * Created on: Jun 16, 2016
5   * Author Christof Pfannenmüller (student06)
6   */
7  #include "Header_general.h" //including all Header files

9  //additional functions
10 void delay(unsigned long delay) {
11     while (delay--) {
12         __NOP();
13     }
14 }

16 //init

18 void init(void) {

20     //sets LED Pins as Outputs
21     XMC_GPIO_SetMode(PORT_LED_1, PIN_LED_1, XMC_GPIO_MODE_OUTPUT_PUSH_PULL); //LED1
22     XMC_GPIO_SetMode(PORT_LED_2, PIN_LED_2, XMC_GPIO_MODE_OUTPUT_PUSH_PULL);
23     XMC_GPIO_SetMode(PORT_LED_3, PIN_LED_3, XMC_GPIO_MODE_OUTPUT_PUSH_PULL);
24     XMC_GPIO_SetMode(PORT_LED_4, PIN_LED_4, XMC_GPIO_MODE_OUTPUT_PUSH_PULL);
25     XMC_GPIO_SetMode(PORT_LED_5, PIN_LED_5, XMC_GPIO_MODE_OUTPUT_PUSH_PULL);
26     XMC_GPIO_SetMode(PORT_LED_6, PIN_LED_6, XMC_GPIO_MODE_OUTPUT_PUSH_PULL);
27     XMC_GPIO_SetMode(PORT_LED_7, PIN_LED_7, XMC_GPIO_MODE_OUTPUT_PUSH_PULL); //LED7
28     //set LED Pins high (active low);
29     XMC_GPIO_SetOutputHigh(PORT_LED_1, PIN_LED_1);
30     XMC_GPIO_SetOutputHigh(PORT_LED_2, PIN_LED_2);
31     XMC_GPIO_SetOutputHigh(PORT_LED_3, PIN_LED_3);
32     XMC_GPIO_SetOutputHigh(PORT_LED_4, PIN_LED_4);
33     XMC_GPIO_SetOutputHigh(PORT_LED_5, PIN_LED_5);
34     XMC_GPIO_SetOutputHigh(PORT_LED_6, PIN_LED_6);
35     XMC_GPIO_SetOutputHigh(PORT_LED_7, PIN_LED_7);

37     //set P_ON Pins as Output
38     XMC_GPIO_SetMode(PORT_P_ON_TDA_1, PIN_P_ON_TDA_1, XMC_GPIO_MODE_OUTPUT_PUSH_PULL);
39     //TDA1
40     XMC_GPIO_SetMode(PORT_P_ON_TDA_2, PIN_P_ON_TDA_2, XMC_GPIO_MODE_OUTPUT_PUSH_PULL);
41     XMC_GPIO_SetMode(PORT_P_ON_TDA_3, PIN_P_ON_TDA_3, XMC_GPIO_MODE_OUTPUT_PUSH_PULL);
42     XMC_GPIO_SetMode(PORT_P_ON_TDA_4, PIN_P_ON_TDA_4, XMC_GPIO_MODE_OUTPUT_PUSH_PULL);
43     XMC_GPIO_SetMode(PORT_P_ON_TDA_5, PIN_P_ON_TDA_5, XMC_GPIO_MODE_OUTPUT_PUSH_PULL);
44     XMC_GPIO_SetMode(PORT_P_ON_TDA_6, PIN_P_ON_TDA_6, XMC_GPIO_MODE_OUTPUT_PUSH_PULL);
45     //TDA6
46     //P_PON low -> TDAs off state
47     XMC_GPIO_SetOutputLow(PORT_P_ON_TDA_1, PIN_P_ON_TDA_1);
48     XMC_GPIO_SetOutputLow(PORT_P_ON_TDA_2, PIN_P_ON_TDA_2);
49     XMC_GPIO_SetOutputLow(PORT_P_ON_TDA_3, PIN_P_ON_TDA_3);
50     XMC_GPIO_SetOutputLow(PORT_P_ON_TDA_4, PIN_P_ON_TDA_4);
51     XMC_GPIO_SetOutputLow(PORT_P_ON_TDA_5, PIN_P_ON_TDA_5);
52     XMC_GPIO_SetOutputLow(PORT_P_ON_TDA_6, PIN_P_ON_TDA_6);

53 }

54 void send_serialnumber_to_com(void) {
55     CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
56     CDC_Device_SendData(&VirtualSerial_CDC_Interface, "Serial Number TDA1\r\n", 20);
57     COM_send_int_as_string(tda5340_get_serial_number(TDA1));
58     CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
59     CDC_Device_SendData(&VirtualSerial_CDC_Interface, "Serial Number TDA2\r\n", 20);
60     COM_send_int_as_string(tda5340_get_serial_number(TDA2));

```

```

60     CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
61     CDC_Device_SendData(&VirtualSerial_CDC_Interface, "Serial Number TDA3\r\n", 20);
62     COM_send_int_as_string(tda5340_get_serial_number(TDA3));
63     CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
64     CDC_Device_SendData(&VirtualSerial_CDC_Interface, "Serial Number TDA4\r\n", 20);
65     COM_send_int_as_string(tda5340_get_serial_number(TDA4));
66     CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
67     CDC_Device_SendData(&VirtualSerial_CDC_Interface, "Serial Number TDA5\r\n", 20);
68     COM_send_int_as_string(tda5340_get_serial_number(TDA5));
69     CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
70     CDC_Device_SendData(&VirtualSerial_CDC_Interface, "Serial Number TDA6\r\n", 20);
71     COM_send_int_as_string(tda5340_get_serial_number(TDA6));
72     CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
73 }
74 void general_test(void) {
75     //uint8_t i = 0;
76     //i=tda5340_transfer(0, 0x05, 0xD0 , 0);
77     //
78     //if(i==0){
79     //    led_on(i);i=0;
80     //}
81     //test für TDA Lib von Felix

83     //tda5340_gpio_init(0);
84     //tda5340_init(0);
85     //uint32_t serialnumber = tda5340_get_serial_number(0);

87     //serialnumber -> LEDs
88     //for (int var = 0; var < 32; var++) {
89     //    if (serialnumber & (1 << var)) {
90     //        led_on(5);
91     //    }
92     //    led_on(6);
93     //    delay(4000000);
94     //    led_off(5);
95     //    led_off(6);
96     //}

98     set_TDA_status(0, 1);
99     delay(40000);

101     spi_init(spi_master_ch);
102     delay(40000);

104     tda5340_transfer(5, READ_FROM_CHIP, IS2, 0xFF);
105     delay(40000);
106     tda5340_transfer(5, READ_FROM_CHIP, 0xDB, 0);

108     tda5340_transfer(5, READ_FROM_CHIP, IS2, 0xFF);
109     delay(40000);
110     tda5340_transfer(5, READ_FROM_CHIP, 0xDB, 0);

112 //
113 //
114 //
115 //     uint16_t spi_array_tx[20] = { 0 };
116 //     spi_array_tx[0] =0x05;
117 //     spi_array_tx[1] =0xD3;
118 //     uint16_t spi_array_rx[20] = { 0 };
119 //     led_on(2);
120 //
121 //
122 //     set_TDA_status(0,1);
123 //
124 //
125 //     led_on(6);
126 //     led_on(7);
127 //

```



```

128 //          if (spi_array_rx[0] == 0 &&spi_array_rx[1] == 0&&spi_array_rx[2] ==
    0&&spi_array_rx[3] == 0&&spi_array_rx[4] == 0&&spi_array_rx[5] == 0
129 //          && spi_array_rx[6] == 0 &&spi_array_rx[7] == 0&&spi_array_rx[7] ==
    0&&spi_array_rx[8] == 0&&spi_array_rx[9] == 0 ) {
130 //              led_off(6);
131 //          }
132 //
133 //
134 // spi_init(spi_master_ch);
135 // spi_transfer(spi_master_ch, 4, spi_array_tx, spi_array_rx, 20);
136 //
137 //
138 //
139 // for (int var = 0; var < 20; var++) {
140 //     if (spi_array_rx[var] != 0) {
141 //         led_on(5);
142 //     }
143 // }

149 // CDC_Device_SendData(&VirtualSerial_CDC_Interface, "Start Reading FIFO TDA 3\r\n
    ", 26);
150 //
151 // int8_t data_send[10] = { 1, 3, 5, 7, 9, 2, 4, 6, 8, 10 };
152 // int8_t data_rec[10];
153 // uint8_t lenght = 10;
154 // tda5340_fifo_rw(TDA3, 1, data_send, &lenght);
155 // tda5340_fifo_rw(TDA3, 0, data_rec, &lenght);
156 // COM_send_int_as_string(data_rec[0]);
157 // CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
158 //
159 // COM_send_int_as_string(data_rec[1]);
160 // CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
161 //
162 // COM_send_int_as_string(data_rec[2]);
163 // CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
164 //
165 // COM_send_int_as_string(data_rec[3]);
166 // CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
167 //
168 // COM_send_int_as_string(data_rec[4]);
169 // CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
170 //
171 // COM_send_int_as_string(data_rec[5]);
172 // CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
173 //
174 // COM_send_int_as_string(data_rec[6]);
175 // CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
176 //
177 // COM_send_int_as_string(data_rec[7]);
178 // CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
179 //
180 // COM_send_int_as_string(data_rec[8]);
181 // CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
182 //
183 // COM_send_int_as_string(data_rec[9]);
184 // CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
185 //
186 // COM_send_int_as_string(data_rec[10]);
187 // CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);
188 //
189 // CDC_Device_SendData(&VirtualSerial_CDC_Interface, "\r\n", 2);

191 CDC_Device_USBTask(&VirtualSerial_CDC_Interface);

193 // set_TDA_status(TDA_ALL, 0);

```

```
194    //  set_TDA_status(TDA_ALL, 1);
195    //
196    //  set_TDA_status(TDA_ALL, 0);
197    //  set_TDA_status(TDA_ALL, 1);
198    //  set_TDA_status(TDA_ALL, 0);
199    //  set_TDA_status(TDA_ALL, 1);
```

```
204 }
```

Literatur
