INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

LISTINGS

ACRONYM

1

BASISSOFTWARE

1.1 SCHICHTENARCHITEKTUR

Die in AUTOSAR definierte Schichtenarchitektur soll eine einfachere Portierung von Software auf unterschiedliche Hardware ermöglichen. Mussten bislang bei ungünstig konzipierten Software-Architekturen verschiedene Stellen bis hin zur Anwendungsschicht umfangreich angepasst werden, müssen mit AUTOSAR lediglich alle mikrocontroller-spezifischen Treiber im MCAL ersetzt werden. Dadurch reduziert sich der Implementierungs- und Testaufwand sowie das damit verbundene Risiko deutlich. Softwarekomponenten können so durch die hardwareunabhängige Schnittstellen leicht auf unterschiedliche Steuergeräte übertragen werden. In AUTOSAR unterscheidet man grundsätzlich die folgende Schichten:

- Anwendungsschicht
- Runtime Environment (RTE)
- Service Layer
- ECU Abstraction Layer
- Microscontroller Abstraction Layer (MCAL)

Die Umsetzung des Projekts erforderte konkrete Implementierungen sowohl im Bereich der Service Layer als auch in der ECU Hardware Abstraction.

1.1.1 *Service Layer*

Die Service Layer enthält im Allgemeinen die Betriebsystem-Funktionen und stellt verschiedene Arten von Hintergrunddiensten wie Netzwerkdienste, Speicherverwaltung und Buskommunikationsdienste für die Anwendungsschicht bereit. Sie ist die höchste Schicht der Basissoftware und ist damit essentiell für die Anwendungsschicht. Die Implementierung ist in den meisten Fällen hardwareunabhängig und damit leicht austauschbar.

1.1.1.1 *Sound Handler*

Der Sound Handler stellt der RTE spezifische Funktionen zur Audiowiedergabe zur Verfügung. Als Funktionalität implementiert der Sound Handler dabei sowohl die einfache Wiedergabe eines Tons oder einer WAV-Datei als auch die Mehrfachwiedergabe mit optionaler Pause. Umgesetzt wird dies durch folgende Methoden:

```
void play_single_wav(U32 freq, U32 ms, U32 vol);
void play_single_wav(U32 file, U32 length, U32 freq, U32 vol);

void play_multiple_tones(U32 freq, U32 ms, U32 vol, U32 rep, U32 pause);
void play_multiple_wavs(const CHAR *file, U32 length, S32 freq, U32 vol, U32 rep, U32
    pause);
```

1.1.1.2 *Motor Handler*

1.1.1.3 *Communication Manager*

1.1.1.4 *Display Handler*

1.1.2 *ECU Hardware Abstraction*

Die ECU Hardware Abstraction Layer versteckt den konkreten Aufbau des Steuergeräts und bietet einen einheitlichen Zugriff auf alle Funktionalitäten eines Steuergeräts wie Kommunikation, Speicher oder IO - unabhängig davon, ob diese Funktionalitäten Bestandteil des Mikrocontrollers sind oder durch Peripheriekomponenten realisiert werden.

1.1.2.1 *Ultraschall Hardware Abstraction*

1.1.2.2 *ADC Hardware Abstraction*

1.1.2.3 *DIO Hardware Abstraction*

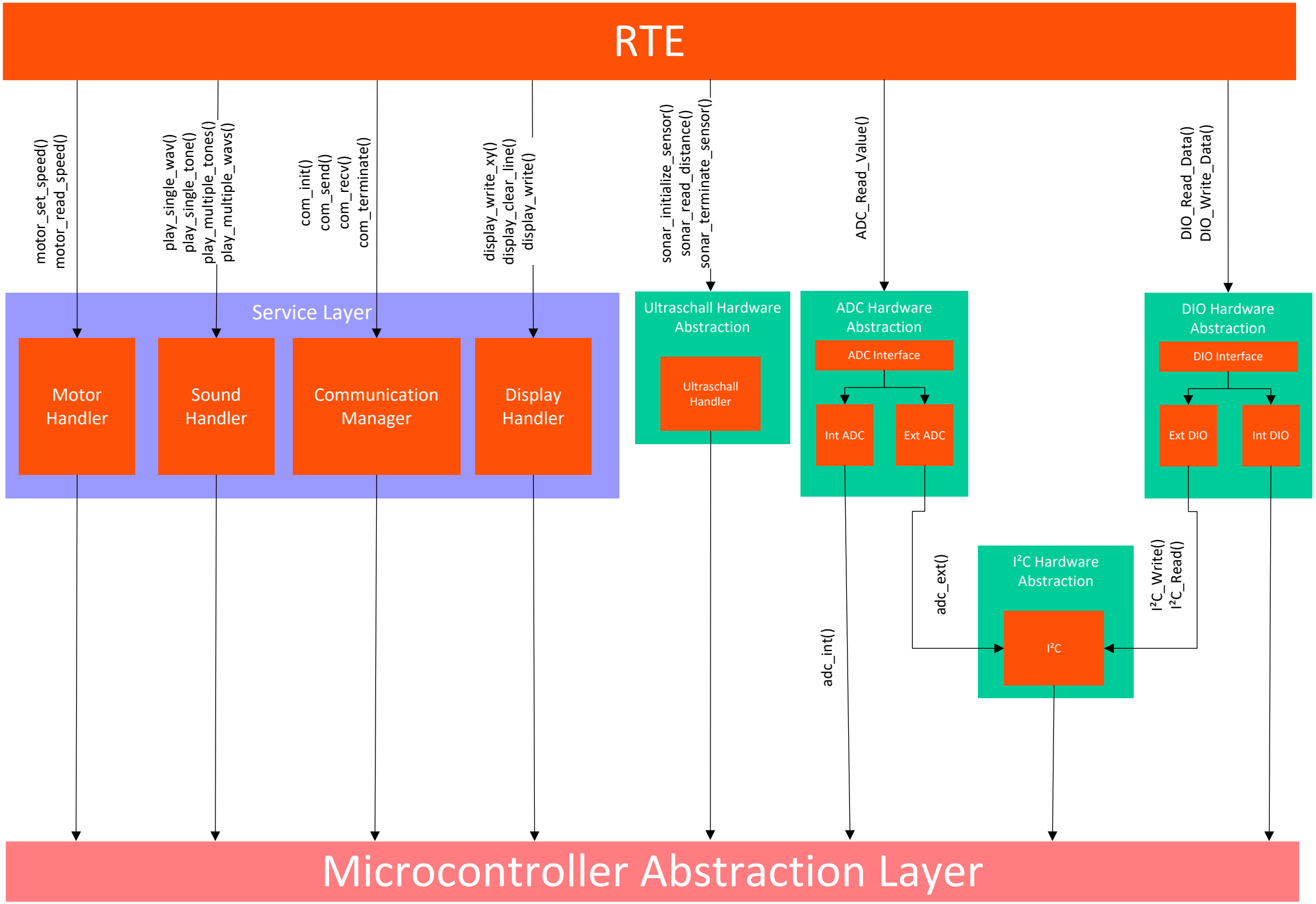
1.1.2.4 *I2C Hardware Abstraction*

2

HARDWAREAUFBAU

A

ANHANG


RTE
motor_set_speed()
motor_read_speed()
play_single_wav()
play_single_tone()
play_multiple_tones()
play_multiple_wavs()
com_init()
com_send()
com_recv()
com_terminate()
display_write_xy()
display_clear_line()
display_write()
sonar_initialize_sensor()
sonar_read_distance()
sonar_terminate_sensor()
ADC_Read_Value()
DIO_Read_Data()
DIO_Write_Data()
Service Layer
Motor Handler
Sound Handler
Communication Manager
Display Handler
Ultraschall Hardware Abstraction
Ultraschall Handler
ADC Hardware Abstraction
ADC Interface
Int ADC
Ext ADC
DIO Hardware Abstraction
DIO Interface
Ext DIO
Int DIO
adc_ext()
adc_int()
I²C Hardware Abstraction
I²C
I²C_Write()
I²C_Read()
Microcontroller Abstraction Layer

Test. Ende im Gelände

LITERATURVERZEICHNIS

[1] ARM: *AMR7TDMI Technical Reference Manual*, 14.06.2018, http://infocenter.arm.com/help/topic/com.arm.doc.ddi0210c/DDI0210B.pdf