

# Entwicklung und Programmierung einer Multifunktions-Adapterplatine für ein mbed-Mikrocontrollermodul

eingereicht am 14.01.2012

von

cand. M.Sc. Maik Gotzmann

Matrikel-Nr.: 6200338

**Gutachter:**

Prof. Dr.-Ing. Bernhard Lampe

Universität Rostock

Richard-Wagner-Straße 31

18119 Rostock-Warnemünde

**Zweitgutachter:**

Dipl.-Ing. Martin Kurowski

Universität Rostock

Richard-Wagner-Straße 31

18119 Rostock-Warnemünde

**Masterarbeit**

**Fakultät für Informatik und Elektrotechnik**

## Master Thesis

für Herrn B. Sc. Maik Gotzmann

### Entwicklung und Programmierung einer Multifunktions- Adapterplatine für ein mbed-Mikrocontrollermodul

Das mbed NXP LPC1768 ist ein Mikrocontrollermodul auf Basis eines LPC1768 96MHz ARM Cortex-M3 Prozessors. Es ist mit seinem 40-Pin DIP-Format, seiner hohen Rechenleistung und der Vielzahl von unterstützten Schnittstellen bestens für die Anwendung im Sensorik-, Automatisierungsbereich geeignet.

Für dieses Mikrocontrollermodul soll innerhalb der Master Thesis eine Adapterplatine entwickelt werden, welche die verschiedenen Schnittstellen physisch zur Verfügung stellt, aber auch Schnittstellenumschaltungen beachtet. Die Stromversorgung soll über das PoE-Verfahren realisiert werden. Als eine Anwendung soll die Platine als Schnittstellenkonverter fungieren, der die vorhandenen Schnittstellen ausliest und die Daten in ein definiertes Ethernetprotokoll umsetzt und so bereitstellt. Zur Realisierung dieser Funktionalitäten ist eine Softwarekonzeption zu erarbeiten und die programmiertechnische Umsetzung durchzuführen. Desweiteren ist eine funktionale grafische Bedienoberfläche (GUI) zur Interaktion mit der entwickelten Platine zu programmieren.

Im Rahmen der Master Thesis sind im Einzelnen folgende Arbeiten durchzuführen:

1. Entwicklung einer Schaltung und Prototypenplatine, welche die physischen Schnittstellen Ethernet, RS232, RS485/RS422, CAN, I<sup>2</sup>C, SPI, USB und Analog-Input modular bereitstellt (galvanische Trennung unter Beachtung IEC61000), Realisierung der Stromversorgung der Baugruppe über das PoE-Verfahren, Bereitstellung einer abgesicherten Stromversorgung für externe Geräte.
2. Konzeptionierung und Programmierung eines Schnittstellenumsetzers, der die Protokolle der verfügbaren Schnittstellen einliest und in ein definiertes Ethernetprotokoll umsetzt.
3. Test der Baugruppe für die Umsetzung von RS232/422 NMEA-Protokollen eines Schiffsführungssimulators in das entwickelte Ethernetprotokoll.
4. Entwicklung einer GUI zur Parametrierung der Platine (z.B. Baudraten, Schnittstellenumschaltungen, etc.).

Der Schaltungsentwurf wird mit der Layout Software EAGLE durchgeführt. Die Programmierung des Moduls erfolgt in C mit der LPCXpresso IDE.

**Tag der Ausgabe:** 15.06.2011

**Tag der Abgabe:** 14.12.2011

**Betreuer:** Dipl.-Ing. Martin Kurowski, Dipl.-Ing. Matthias Wulff

**Erstgutachter:** Prof. Dr.-Ing. Bernhard Lampe

**Zweitgutachter:** Dipl.-Ing. Martin Kurowski

#### INSTITUT FÜR AUTOMATISIERUNGSTECHNIK

# Inhaltsverzeichnis

|          |                                                              |           |
|----------|--------------------------------------------------------------|-----------|
| <b>1</b> | <b>Einleitung</b>                                            | <b>1</b>  |
| 1.1      | Motivation und Zielsetzung . . . . .                         | 1         |
| 1.2      | Aufgabenstellung . . . . .                                   | 1         |
| 1.3      | Aufbau der Masterarbeit . . . . .                            | 2         |
| <b>2</b> | <b>Grundlagen zu den physischen Schnittstellen</b>           | <b>4</b>  |
| 2.1      | Die klassischen seriellen Peripheralschnittstellen . . . . . | 4         |
| 2.1.1    | Erzeugung des Datenpaketes auf Logikebene . . . . .          | 4         |
| 2.1.2    | RS-232 Interface . . . . .                                   | 5         |
| 2.1.3    | RS-422 Interface . . . . .                                   | 7         |
| 2.1.4    | RS-485 Interface . . . . .                                   | 11        |
| 2.2      | Nahbereichskommunikation auf Platineebene . . . . .          | 12        |
| 2.2.1    | Serial Peripheral Interface . . . . .                        | 12        |
| 2.2.2    | Inter Integrated Circuit Bus . . . . .                       | 14        |
| 2.3      | Universal Serial Bus . . . . .                               | 16        |
| 2.4      | Controller Area Network . . . . .                            | 18        |
| 2.5      | Ethernet Interface . . . . .                                 | 19        |
| <b>3</b> | <b>Konzept der Multifunktions-Adapterplatine</b>             | <b>22</b> |
| 3.1      | Hardwarekonzept . . . . .                                    | 22        |
| 3.1.1    | Grundkonzept . . . . .                                       | 22        |
| 3.1.2    | Das <i>mbed</i> -Mikrocontrollermodul . . . . .              | 23        |
| 3.1.3    | Das Base-Board . . . . .                                     | 25        |
| 3.1.4    | Modularisierung durch Breakout-Boards . . . . .              | 27        |
| 3.1.5    | Spannungsversorgung durch Power over Ethernet . . . . .      | 29        |
| 3.1.6    | Galvanische Trennung . . . . .                               | 31        |
| 3.1.7    | Überspannungsschutz . . . . .                                | 33        |
| 3.2      | Softwarekonzept . . . . .                                    | 35        |
| 3.2.1    | Grundkonzept . . . . .                                       | 35        |
| 3.2.2    | Hardwareschicht . . . . .                                    | 37        |
| 3.2.3    | Das Konfigurationsmodul . . . . .                            | 40        |
| 3.2.4    | uIP-Stack und Webserver . . . . .                            | 42        |
| 3.2.5    | Das FIFO-Modul . . . . .                                     | 43        |
| 3.2.6    | Das Service-Modul . . . . .                                  | 46        |

|          |                                                                            |           |
|----------|----------------------------------------------------------------------------|-----------|
| <b>4</b> | <b>Hardware-Entwurf und Realisierung</b>                                   | <b>49</b> |
| 4.1      | Allgemeines . . . . .                                                      | 49        |
| 4.2      | Aufbau des Base-Boards . . . . .                                           | 50        |
| 4.2.1    | Spannungsversorgung . . . . .                                              | 50        |
| 4.2.2    | Schnittstellen . . . . .                                                   | 52        |
| 4.3      | Aufbau der Breakout-Boards . . . . .                                       | 55        |
| 4.3.1    | Aufbau des RS-232 Breakout-Boards . . . . .                                | 55        |
| 4.3.2    | Aufbau des RS-422 Breakout-Boards . . . . .                                | 56        |
| 4.3.3    | Aufbau des RS-485 Breakout-Boards . . . . .                                | 57        |
| 4.3.4    | Aufbau des EEPROM Breakout-Boards . . . . .                                | 57        |
| <b>5</b> | <b>Implementierung der mbed-Software</b>                                   | <b>58</b> |
| 5.1      | Aufbau des Softwareprojektes und Hauptprogrammablauf . . . . .             | 58        |
| 5.2      | Implementierung der Hardwareerschicht . . . . .                            | 59        |
| 5.2.1    | Der ADC-Treiber . . . . .                                                  | 59        |
| 5.2.2    | Der CAN-Treiber . . . . .                                                  | 60        |
| 5.2.3    | Der EEPROM-Treiber . . . . .                                               | 61        |
| 5.2.4    | Der $I^2C$ -Treiber . . . . .                                              | 63        |
| 5.2.5    | Der SPI-Treiber . . . . .                                                  | 64        |
| 5.2.6    | Der UART-Treiber . . . . .                                                 | 65        |
| 5.2.7    | Der GPIO- und TIMER-Treiber . . . . .                                      | 65        |
| 5.3      | Implementierung der Applikationsschicht . . . . .                          | 66        |
| 5.3.1    | Das Konfigurationsmodul . . . . .                                          | 66        |
| 5.3.2    | Das FIFO-Modul . . . . .                                                   | 67        |
| 5.3.3    | Integration des uIP-Stack . . . . .                                        | 69        |
| <b>6</b> | <b>Implementierung des Webserver und Konfigurationsmöglichkeiten</b>       | <b>71</b> |
| 6.1      | Die Webserver Applikation . . . . .                                        | 71        |
| 6.2      | Aufbau der dynamischen Webseiten . . . . .                                 | 71        |
| 6.3      | Konfigurationsmöglichkeiten durch den Webserver . . . . .                  | 73        |
| 6.3.1    | Grundlegende Funktionen . . . . .                                          | 73        |
| 6.3.2    | Konfiguration der auf UART basierenden Schnittstellen . . . . .            | 74        |
| 6.3.3    | Konfiguration der SPI-Schnittstelle . . . . .                              | 75        |
| 6.3.4    | Konfiguration der $I^2C$ -Schnittstelle . . . . .                          | 76        |
| 6.3.5    | Konfiguration der CAN-Schnittstelle . . . . .                              | 77        |
| 6.3.6    | Konfiguration der Netzwerkeinstellungen . . . . .                          | 78        |
| 6.3.7    | Konfiguration des ADC . . . . .                                            | 79        |
| 6.3.8    | Konfiguration der GPIO-Funktionalität . . . . .                            | 79        |
| <b>7</b> | <b>Test der Hardwareplattform und Validierung der Software</b>             | <b>80</b> |
| 7.1      | Allgemeine Funktionsprüfung . . . . .                                      | 80        |
| 7.2      | Validierung der Baudrate der auf UART basierenden Schnittstellen . . . . . | 81        |
| 7.3      | Test der Gesamtfunktionalität . . . . .                                    | 82        |

|          |                                                                      |            |
|----------|----------------------------------------------------------------------|------------|
| 7.4      | Test der CAN-Schnittstelle . . . . .                                 | 84         |
| 7.5      | Test mit Hilfe des Schiffsführungssimulators . . . . .               | 85         |
| <b>8</b> | <b>Zusammenfassung und Ausblick</b>                                  | <b>87</b>  |
| 8.1      | Zusammenfassung . . . . .                                            | 87         |
| 8.2      | Ausblick . . . . .                                                   | 88         |
|          | <b>Anhang</b>                                                        | <b>88</b>  |
| <b>A</b> | <b><i>mbed</i> &amp; Breakout-Socket Mapping</b>                     | <b>89</b>  |
| <b>B</b> | <b>Technische Details zum Base-Board</b>                             | <b>91</b>  |
| B.1      | Technische Spezifikation . . . . .                                   | 91         |
| B.2      | Schaltplan . . . . .                                                 | 92         |
| B.3      | Layout und Bestückungsplan . . . . .                                 | 97         |
| B.4      | Bestückungsliste . . . . .                                           | 99         |
| <b>C</b> | <b>Technische Details zum RS-232 Breakout-Board</b>                  | <b>102</b> |
| C.1      | Technische Spezifikation . . . . .                                   | 102        |
| C.2      | Schaltplan . . . . .                                                 | 103        |
| C.3      | Layout und Bestückungsplan . . . . .                                 | 104        |
| C.4      | Bestückungsliste . . . . .                                           | 105        |
| <b>D</b> | <b>Konzeptionelle Details zum RS-422 Breakout-Board</b>              | <b>106</b> |
| D.1      | Technische Spezifikation . . . . .                                   | 106        |
| D.2      | Schaltplan . . . . .                                                 | 107        |
| D.3      | Layout und Bestückungsplan . . . . .                                 | 108        |
| D.4      | Bestückungsliste . . . . .                                           | 109        |
| <b>E</b> | <b>Konzeptionelle Details zum RS-485 Breakout-Board</b>              | <b>111</b> |
| E.1      | Technische Spezifikation . . . . .                                   | 111        |
| E.2      | Schaltplan . . . . .                                                 | 112        |
| E.3      | Layout und Bestückungsplan . . . . .                                 | 113        |
| E.4      | Bestückungsliste . . . . .                                           | 114        |
| <b>F</b> | <b>Technische Details zum EEPROM-Breakout-Board</b>                  | <b>116</b> |
| F.1      | Technische Spezifikation . . . . .                                   | 116        |
| F.2      | Schaltplan . . . . .                                                 | 116        |
| <b>G</b> | <b>Übersicht aller Konfigurationsbefehle</b>                         | <b>117</b> |
| G.1      | Konfigurationsbefehle der Hardwaregruppe <i>mbed</i> . . . . .       | 117        |
| G.2      | Konfigurationsbefehle der Hardwaregruppe Breakout-Socket 1 . . . . . | 118        |
| G.3      | Konfigurationsbefehle der Hardwaregruppe Breakout-Socket 2 . . . . . | 119        |
| G.4      | Konfigurationsbefehle der Hardwaregruppe Breakout-Socket 3 . . . . . | 120        |
| G.5      | Konfigurationsbefehle der Hardwaregruppe Breakout-Socket 4 . . . . . | 121        |

|      |                                                                     |     |
|------|---------------------------------------------------------------------|-----|
| G.6  | Konfigurationsbefehle der Hardwaregruppe CAN . . . . .              | 122 |
| G.7  | Konfigurationsbefehle der Hardwaregruppe ETHERNET . . . . .         | 123 |
| G.8  | Konfigurationsbefehle der Hardwaregruppe Extended ETHERNET2 . . . . | 123 |
| G.9  | Konfigurationsbefehle der Hardwaregruppe Extended CAN2 . . . . .    | 124 |
| G.10 | Konfigurationsbefehle der Hardwaregruppe Extended CAN3 . . . . .    | 124 |
| G.11 | Konfigurationsbefehle der Hardwaregruppe Extended Others . . . . .  | 125 |

## Abkürzungsverzeichnis

|               |                                                                |
|---------------|----------------------------------------------------------------|
| ACK .....     | Acknowledge                                                    |
| ADC .....     | Analog Digital Converter                                       |
| API .....     | Application Programming Interface                              |
| ARP .....     | Address Resolution Protocol                                    |
| ASCII .....   | American Standard Code for Information Interchange             |
| CAN .....     | Controller Area Network                                        |
| CCITT .....   | Comité Consultatif International Téléphonique et Télégraphique |
| CLI .....     | Command Line Interface                                         |
| CMOS .....    | Complementary Metal Oxide Semiconductor                        |
| CMSIS .....   | Cortex Microcontroller Software Interface Standard             |
| CPHA .....    | Clock Phase                                                    |
| CPOL .....    | Clock Polarity                                                 |
| CPU .....     | Central Processing Unit                                        |
| CRC .....     | Cyclic Redundancy Check                                        |
| CS .....      | Chip Select                                                    |
| CSMA/CD ..... | Carrier Sense Multiple Access / Collision Detection            |
| DCE .....     | Data Circuit-Terminating Equipment                             |
| DIP .....     | Dual In-line Package                                           |
| DMA .....     | Direct Memory Access                                           |
| DTE .....     | Data Terminal Equipment                                        |
| EEPROM .....  | Electrically Erasable Programmable Read-Only Memory            |
| EFT .....     | Electrical Fast Transient/Burst                                |
| EIA .....     | Electronic Industries Alliance                                 |
| EMV .....     | Elektromagnetische Verträglichkeit                             |
| ESD .....     | Electrostatic Discharge                                        |
| FIFO .....    | First In - First Out                                           |
| GPIO .....    | General Purpose Input/Output                                   |
| GUI .....     | Graphical User Interface                                       |
| HTML .....    | HyperText Markup Language                                      |

|                        |                                                   |
|------------------------|---------------------------------------------------|
| HTTP .....             | Hypertext Transfer Protocol                       |
| I <sup>2</sup> C ..... | Inter Integrated Circuit                          |
| IC .....               | Integrated Circuit                                |
| ICMP .....             | Internet Control Message Protocol                 |
| ID .....               | Identifikator                                     |
| IDE .....              | Integrated Development Environment                |
| IEC .....              | International Electrotechnical Commission         |
| IEEE .....             | Institute of Electrical and Electronics Engineers |
| IP .....               | Internet-Protokoll                                |
| ISR .....              | Interrupt Service Routine                         |
| ITU .....              | International Telecommunication Union             |
| JTAG .....             | Joint Test Action Group                           |
| KiB .....              | Kibibyte = 1024 Byte                              |
| LAN .....              | Local Area Network                                |
| LSB .....              | Least Significant Bit                             |
| MAC .....              | Media Access Control                              |
| MCU .....              | Mikrocontroller Unit                              |
| MII .....              | Media Independent Interface                       |
| MISO .....             | Master In Slave Out                               |
| MLT-3 .....            | Multilevel Transmission Encoding - 3 levels       |
| MOSI .....             | Master Out Slave In                               |
| MSB .....              | Most Significant Bit                              |
| MTU .....              | Maximum Transmission Unit                         |
| NACK .....             | Not Acknowledge                                   |
| NMEA .....             | National Marine Electronics Association           |
| NRZI .....             | Non Return to Zero Invert                         |
| OSI .....              | Open Systems Interconnection Reference Model      |
| PC .....               | Personal Computer                                 |
| PCB .....              | Printed Circuit Board                             |
| PD .....               | Powered Device                                    |
| PoE .....              | Power over Ethernet                               |
| PSE .....              | Power Sourcing Equipment                          |
| RTS .....              | Ready to Send                                     |
| SCL .....              | Serial Clock Line                                 |
| SCLK .....             | SPI Clock                                         |

|            |                                             |
|------------|---------------------------------------------|
| SDA .....  | Serial Data Line                            |
| SFD .....  | Start Frame Delimiter                       |
| SMD .....  | Surface-Mounted Device                      |
| SPI .....  | Serial Peripheral Interface                 |
| SRAM ..... | Static Random Access Memory                 |
| SSI .....  | Server Side Includes                        |
| SSP .....  | Synchronous Serial Port                     |
| TCP .....  | Transmission Control Protocol               |
| TTL .....  | Transistor-Transistor-Logik                 |
| TVS .....  | Transient Voltage Suppressor                |
| UART ..... | Universal Asynchronous Receiver Transmitter |
| UDP .....  | User Datagram Protocol                      |
| UL .....   | Unit Load                                   |
| USB .....  | Universal Serial Bus                        |
| UTP .....  | Unshielded Twisted Pair                     |
| VLAN ..... | Virtual Local Area Network                  |
| VoIP ..... | Voice over IP                               |
| WLAN ..... | Wireless Local Area Network                 |
| WWW .....  | World Wide Web                              |

## Abbildungsverzeichnis

|      |                                                                                             |    |
|------|---------------------------------------------------------------------------------------------|----|
| 2.1  | Erzeugter Logikpegel durch den UART in 8-N-1 Codierung . . . . .                            | 5  |
| 2.2  | Umsetzung des UART-Pegels in RS-232 Spannungspegel . . . . .                                | 7  |
| 2.3  | Verbindung zweier DTE über 9pol. D-SUB Verbindung . . . . .                                 | 8  |
| 2.4  | Symmetrische Signalübertragung und Differenzsignalbildung . . . . .                         | 9  |
| 2.5  | RS-422 Bussystem in Full-Duplex Konfiguration . . . . .                                     | 10 |
| 2.6  | RS-485 Bussystem in Halb-Duplex Konfiguration . . . . .                                     | 12 |
| 2.7  | Beispiel einer 16-Bit SPI-Kommunikation im Modus 0 . . . . .                                | 13 |
| 2.8  | I <sup>2</sup> C-Bus mit mehreren Mastern . . . . .                                         | 14 |
| 2.9  | Leitungscodierung mit Bit Stuffing und NRZI . . . . .                                       | 17 |
| 2.10 | Wandlung des Logik-Pegels zum CAN-Leitungsspannungspegel . . . . .                          | 18 |
| 2.11 | Aufbau des Ethernet-Datenpaketes . . . . .                                                  | 21 |
| 3.1  | Kontaktbelegung des <i>mbed</i> -Mikrocontrollermoduls . . . . .                            | 24 |
| 3.2  | Schematischer Aufbau des Base-Boards . . . . .                                              | 25 |
| 3.3  | CAN-Bus in Daisy-Chain Bustopologie . . . . .                                               | 26 |
| 3.4  | Generischer Aufbau eines Breakout-Boards . . . . .                                          | 28 |
| 3.5  | Spannungsversorgung eines PD durch ein PSE . . . . .                                        | 30 |
| 3.6  | Blockschaltbild eines PoE-Moduls . . . . .                                                  | 31 |
| 3.7  | Grundschemata der Isolation der Systemkomponente von der Schnittstellenkomponente . . . . . | 33 |
| 3.8  | Kennlinienverlauf einer unidirektionalen TVS-Diode . . . . .                                | 34 |
| 3.9  | Modell der <i>mbed</i> -Software . . . . .                                                  | 36 |
| 3.10 | Aufbau des 16-Bit ADC-Datenwortes . . . . .                                                 | 39 |
| 3.11 | Aufbau und Struktur der Konfigurationsbefehle . . . . .                                     | 41 |
| 3.12 | Aufbau eines UDP-Paketes auf IPv4 Ebene . . . . .                                           | 43 |
| 3.13 | Exemplarischer Aufbau der UDP-Nutzdaten . . . . .                                           | 45 |
| 3.14 | In-Operation Monitor in der Terminalemulation . . . . .                                     | 46 |
| 3.15 | Ausschnitt aus dem MAP File . . . . .                                                       | 47 |
| 4.1  | Die beiden Base-Boards im bestückten Zustand . . . . .                                      | 49 |
| 4.2  | Komponenten des Base-Boards in der Übersicht . . . . .                                      | 50 |
| 4.3  | Schema der dualen Spannungsversorgung des Base-Boards . . . . .                             | 51 |
| 4.4  | Schematische Darstellung der CAN-Schnittstelle . . . . .                                    | 54 |
| 4.5  | Realisiertes Breakout-Board der RS-232 Schnittstelle . . . . .                              | 56 |
| 5.1  | Projektstruktur der <i>mbed</i> -Software . . . . .                                         | 58 |
| 5.2  | Speicherbelegung des Akzeptanzfilter RAM . . . . .                                          | 61 |

|      |                                                                            |     |
|------|----------------------------------------------------------------------------|-----|
| 5.3  | Ablauf der Autoerkennung eines validen EEPROM-Speichers . . . . .          | 62  |
| 5.4  | $I^2C$ -Kommunikation zwischen Master und Slave als Zustandsmaschine . .   | 64  |
| 5.5  | Aufbau und Struktur eines FIFO-Speichers . . . . .                         | 68  |
| 6.1  | Ein- und Ausgabe durch eine dynamische Webseite . . . . .                  | 72  |
| 6.2  | Webserver-Dialog zur Verwaltung der Gesamtkonfiguration . . . . .          | 74  |
| 6.3  | Webserver-Dialog zur Auswahl des Betriebsmodus . . . . .                   | 74  |
| 6.4  | Webserver-Dialog zur UART-Konfiguration . . . . .                          | 75  |
| 6.5  | Webserver-Dialog zur SPI-Konfiguration . . . . .                           | 76  |
| 6.6  | Webserver-Dialog zur $I^2C$ -Konfiguration . . . . .                       | 76  |
| 6.7  | Webserver-Dialog zur generellen Konfiguration der CAN-Schnittstelle . . .  | 77  |
| 6.8  | Webserver-Dialog zur Konfiguration des CAN Gruppen-ID Akzeptanzfilters     | 77  |
| 6.9  | Webserver-Dialog zur Server-Netzwerkeinstellung . . . . .                  | 78  |
| 6.10 | Webserver-Dialog zur Client-Netzwerkeinstellung . . . . .                  | 78  |
| 6.11 | Webserver-Dialog zur Ansteuerung des ADC . . . . .                         | 79  |
| 6.12 | Webserver-Dialog zur Steuerung der GPIO-Funktionalität . . . . .           | 79  |
| 7.1  | Software <i>Wireshark</i> zur Analyse der versendeten UDP-Pakete . . . . . | 80  |
| 7.2  | Software <i>HTerm</i> als RS-232 Testsender . . . . .                      | 82  |
| 7.3  | PERL-Script als Testempfänger . . . . .                                    | 83  |
| 7.4  | Dialog der Software <i>PCAN-View</i> . . . . .                             | 84  |
| A.1  | <i>mbed</i> & Breakout-Socket 1 & 2 Mapping . . . . .                      | 89  |
| A.2  | <i>mbed</i> & Breakout-Socket 3 & 4 Mapping . . . . .                      | 90  |
| B.1  | Schaltplan des Base-Boards (Seite 1/5) . . . . .                           | 92  |
| B.2  | Schaltplan des Base-Boards (Seite 2/5) . . . . .                           | 93  |
| B.3  | Schaltplan des Base-Boards (Seite 3/5) . . . . .                           | 94  |
| B.4  | Schaltplan des Base-Boards (Seite 4/5) . . . . .                           | 95  |
| B.5  | Schaltplan des Base-Boards (Seite 5/5) . . . . .                           | 96  |
| B.6  | Multi IF Base Board (Top View) . . . . .                                   | 97  |
| B.7  | Multi IF Base Board (Buttom View) . . . . .                                | 98  |
| C.1  | Schaltplan des RS-232 Breakout-Boards . . . . .                            | 103 |
| C.2  | RS-232 Breakout-Board (Top View) . . . . .                                 | 104 |
| C.3  | RS-232 Breakout-Board (Buttom View) . . . . .                              | 104 |
| D.1  | Schaltplan des RS-422 Breakout-Boards . . . . .                            | 107 |
| D.2  | RS-422 Breakout-Board (Top View) . . . . .                                 | 108 |
| D.3  | RS-422 Breakout-Board (Buttom View) . . . . .                              | 108 |
| E.1  | Schaltplan des RS-485 Breakout-Boards . . . . .                            | 112 |
| E.2  | RS-485 Breakout-Board (Top View) . . . . .                                 | 113 |
| E.3  | RS-485 Breakout-Board (Buttom View) . . . . .                              | 113 |

|     |                                                 |     |
|-----|-------------------------------------------------|-----|
| F.1 | Schaltplan des EEPROM-Breakout-Boards . . . . . | 116 |
|-----|-------------------------------------------------|-----|

## Tabellenverzeichnis

|     |                                                                            |     |
|-----|----------------------------------------------------------------------------|-----|
| 2.1 | SPI-Übertragungsmodi . . . . .                                             | 14  |
| 2.2 | I <sup>2</sup> C-Übertragungsmodi . . . . .                                | 15  |
| 3.1 | PD Leistungsklassen . . . . .                                              | 31  |
| 3.2 | UART Testfunktionen des Servicemoduls . . . . .                            | 47  |
| 7.1 | Messwerte zur Überprüfung der tatsächlich erzeugten Baudrate . . . . .     | 81  |
| B.1 | Technische Daten des Base-Boards . . . . .                                 | 91  |
| B.2 | Kontaktierung der D-SUB Verbindung der CAN-Schnittstelle . . . . .         | 91  |
| B.3 | Bestückungsliste Base-Board (Seite 1/3) . . . . .                          | 99  |
| B.4 | Bestückungsliste Base-Board (Seite 2/3) . . . . .                          | 100 |
| B.5 | Bestückungsliste Base-Board (Seite 3/3) . . . . .                          | 101 |
| C.1 | Technische Daten des Breakout-Boards RS-232 . . . . .                      | 102 |
| C.2 | Kontaktierung des RS-232 Breakout-Boards . . . . .                         | 102 |
| C.3 | Bestückungsliste RS-232 Breakout-Board . . . . .                           | 105 |
| D.1 | Technische Daten des Breakout-Boards RS-422 . . . . .                      | 106 |
| D.2 | Kontaktierung des RS-422 Breakout-Boards . . . . .                         | 106 |
| D.3 | Bestückungsliste RS-422 Breakout-Board (Seite 1/2) . . . . .               | 109 |
| D.4 | Bestückungsliste RS-422 Breakout-Board (Seite 2/2) . . . . .               | 110 |
| E.1 | Technische Daten des Breakout-Boards RS-485 . . . . .                      | 111 |
| E.2 | Kontaktierung des RS-485 Breakout-Boards . . . . .                         | 111 |
| E.3 | Bestückungsliste RS-485 Breakout-Board (Seite 1/2) . . . . .               | 114 |
| E.4 | Bestückungsliste RS-485 Breakout-Board (Seite 2/2) . . . . .               | 115 |
| F.1 | Technische Daten des EEPROM-Breakout-Boards . . . . .                      | 116 |
| G.1 | Konfigurationsbefehle der Hardwaregruppe 0h ( <i>mbed</i> ) . . . . .      | 117 |
| G.2 | Konfigurationsbefehle der Hardwaregruppe 1h (Breakout-Socket 1) . . . . .  | 118 |
| G.3 | Konfigurationsbefehle der Hardwaregruppe 2h (Breakout-Socket 2) . . . . .  | 119 |
| G.4 | Konfigurationsbefehle der Hardwaregruppe 3h (Breakout-Socket 3) . . . . .  | 120 |
| G.5 | Konfigurationsbefehle der Hardwaregruppe 4h (Breakout-Socket 4) . . . . .  | 121 |
| G.6 | Konfigurationsbefehle der Hardwaregruppe 5h (CAN) . . . . .                | 122 |
| G.7 | Konfigurationsbefehle der Hardwaregruppe 6h (ETHERNET) . . . . .           | 123 |
| G.8 | Konfigurationsbefehle der Hardwaregruppe Extended 8h (ETHERNET2) . . . . . | 123 |
| G.9 | Konfigurationsbefehle der Hardwaregruppe Extended 9h (CAN2) . . . . .      | 124 |

---

|                                                                              |     |
|------------------------------------------------------------------------------|-----|
| G.10 Konfigurationsbefehle der Hardwaregruppe Extended Ah (CAN3) . . . . .   | 124 |
| G.11 Konfigurationsbefehle der Hardwaregruppe Extended Bh (Others) . . . . . | 125 |

# 1 Einleitung

## 1.1 Motivation und Zielsetzung

Schnittstellen und Bussysteme verbinden elektronische Komponenten auf allen Gebieten der Automatisierungs- und Steuerungstechnik, der Mess- und Regeltechnik sowie in der Datenverarbeitung. Durch die Fortschritte der digitalen Kommunikationstechnologie wurden neue Anwendungsbereiche erschlossen. Durch die rasante Weiterentwicklung der Mikroelektronik kam es zur Ausprägung einer Vielzahl unterschiedlichster Schnittstellentypen, die sich in den Parametern der Übertragungsgeschwindigkeit, Logik- und Leitungsspannungspegel deutlich unterscheiden [12].

Das Betreiben verschiedenster Kommunikationsteilnehmer in einem gemeinsamen Bussystem wird durch die Ausprägung der unterschiedlichen Schnittstellenstandards verhindert. Durch die Industrie werden zwar Schnittstellenwandler für den Einzelfall angeboten, diese Geräte sind jedoch meist an einen speziellen Einsatzzweck gebunden. Die fehlende Flexibilität zur Parametrisierung dieser Schnittstellenwandler grenzt den Anwendungsbereich weiter ein.

Ein Schritt, die Kommunikationsteilnehmer unterschiedlicher Schnittstellen über ein gemeinsames Übertragungsmedium zu vereinen, ist die Entwicklung und der Aufbau einer Multifunktions-Adapterplatine. Dieser multifunktionale Schnittstellenwandler stellt die unterschiedlichen physischen Schnittstellen zum Anschluss von Sensoren und Aktoren aus dem Bereich der Sensorik und Automatisierungstechnik bereit. Der Datenstrom dieser Geräte ist zu erfassen und über ein zentrales Übertragungsmedium zu versenden. Die Bildung dieses Kommunikationsnetzes ermöglicht das Zusammenfassen von Geräten mit unterschiedlichen Schnittstellen und der Versand der Daten kann in einem einheitlichen Protokoll zuverlässig über weite Strecken erfolgen. Der gebündelte Datenstrom kann dann über eine zentrale Schnittstelle, beispielsweise Ethernet, übertragen werden.

Der Konzeption und Entwicklung der Multifunktions-Adapterplatine widmet sich der Inhalt dieser Masterarbeit.

## 1.2 Aufgabenstellung

Das Konzept zur Realisierung der Multifunktions-Adapterplatine basiert auf den Einsatz einer Mikrocontroller Unit (MCU). Dadurch wird ein Höchstmaß an Flexibilität und Parametrisierung der Konfiguration der einzelnen Schnittstellentypen gewährleistet.

Das zur Verfügung stehende *mbed*-Mikrocontrollermodul enthält die MCU *LPC1768* der Firma NXP, der auf den ARM Cortex-M3 Kern basiert. Als zentrale Einheit stellt das *mbed*-Mikrocontrollermodul ausreichend Rechenleistung und die Unterstützung einer Vielzahl von integrierten Schnittstellenfunktionalitäten zur Verfügung. Diese Basis ermöglicht die physische Realisierung folgender Schnittstellentypen auf der Multifunktions-Adapterplatine:

- Ethernet
- auf Universal Asynchronous Receiver Transmitter (UART) basierende Schnittstellen
- Controller Area Network (CAN)
- Serial Peripheral Interface (SPI)
- Inter Integrated Circuit Bus (I<sup>2</sup>C)
- Universal Serial Bus (USB)
- Analog Digital Converter (ADC)

Um die Anzahl der notwendigen Leitungen innerhalb des Kommunikationsnetzes zu verringern, wird die Spannungsversorgung der Multifunktions-Adapterplatine durch das Verfahren des Power over Ethernet (PoE) erfolgen. Das Power Sourcing Equipment (PSE) speist dazu eine Spannung direkt in das Kabel der Ethernetleitung ein. Das PoE-Modul setzt diese Spannung in die Versorgungsspannung zum Betrieb der Multifunktions-Adapterplatine als Powered Device (PD) um [18].

Unter Berücksichtigung des Standards IEC61000 der International Electrotechnical Commission (IEC) werden die elektronischen Komponenten so gewählt und beschaltet, dass ein zuverlässiger Betrieb der Multifunktions-Adapterplatine auch unter erschwerten Bedingungen gewährleistet ist. Die Maßnahmen zur Sicherstellung der elektromagnetischen Verträglichkeit (EMV) werden durch die galvanische Trennung von Schnittstellenkomponenten und den Schutz der Datenleitungen durch Transient Voltage Suppressor (TVS)-Dioden erreicht.

Die in der Programmiersprache C zu implementierende *mbed*-Software stellt die flexible Konfiguration und den zuverlässigen Betrieb der Funktionalitäten der Multifunktions-Adapterplatine sicher. Die zur Konfiguration notwendige Interaktion zwischen Nutzer und der Hardware wird dabei durch ein Graphical User Interface (GUI) erfolgen.

### 1.3 Aufbau der Masterarbeit

Zu Beginn dieser Masterarbeit werden die Grundlagen der einzelnen zu realisierenden Schnittstellen erläutert. In diesem Kapitel wird die Funktionsweise jeder Schnittstelle auf Ebene des Leitungsspannungspegels und Logikpegels erläutert. Das Verständnis dieser

Informationen bildet die Grundlage zur Realisierung der physischen Schnittstelle und zur Programmierung der entsprechenden Schnittstellentreiber der *mbed*-Software.

Das darauffolgende Kapitel schildert das Gesamtkonzept der Multifunktions-Adapterplatine. Dieser Abschnitt unterteilt sich dabei in die beiden Bereiche Hardwarekonzept und Softwarekonzept. Das Hardwarekonzept beschreibt den physischen Aufbau und die Wechselwirkung der einzelnen elektronischen Komponenten zueinander. Das Softwarekonzept erläutert den Aufbau und die Wirkung der *mbed*-Software auf die Funktionalität der Multifunktions-Adapterplatine.

Das nächste Kapitel beschreibt die Realisierung der Multifunktions-Adapterplatine auf Hardwareebene. Hier erfolgt eine detaillierte Beschreibung der Konstruktion der elektronischen Schaltungen und der verwendeten Bauelemente. Die Darstellung der physischen Realisierung der Multifunktions-Adapterplatine unterteilt sich dabei in die Komponenten Base-Board und den modularen Breakout-Boards.

Ein weiteres Kapitel beschreibt die Implementierung der *mbed*-Software. In diesem Abschnitt werden die einzelnen Softwaremodule vorgestellt und detailliert beschrieben. Dieses Kapitel ist in zwei Abschnitte unterteilt. Der erste Teil beschreibt die Softwarekomponenten, die direkt Einfluss auf die Steuerung und Konfiguration der Hardware der MCU und der Multifunktions-Adapterplatine nehmen. Der zweite Abschnitt gibt detailliert Auskunft über die Implementierung der darauf aufbauenden Softwareschichten.

Der Implementierung der GUI zur Konfiguration der Multifunktions-Adapterplatine ist ein eigenes Kapitel gewidmet. In diesem Abschnitt wird die softwaretechnische Realisierung des Webserver beschrieben. Zur Integration der Webserver-Applikation, den Aufbau der dynamischen Webseiten und zu den Möglichkeiten der Konfiguration wird in diesem Abschnitt detailliert eingegangen.

Den Nachweis zur ordnungsgemäßen Funktion der Hard- und Software der Multifunktions-Adapterplatine soll das darauffolgende Kapitel erbringen. In diesem Abschnitt sind Tests und die Testergebnisse aufgeführt, die die Validität der konstruierten Hardware und implementierten *mbed*-Software verdeutlichen. Die in diesen Tests aufgetretenen Probleme werden ebenso an dieser Stelle erörtert.

In dem letzten und abschließenden Kapitel erfolgt in Form einer Zusammenfassung das Resümee, inwiefern die gesteckten Ziele mit dem Ergebnis dieser Masterarbeit eine Konvergenz aufweisen. Der in diesem Abschnitt dargelegte Ausblick gibt Verbesserungen und Erweiterungen zur Gesamtfunktionalität der Multifunktions-Adapterplatine wieder.

Der Inhalt des Anhangs enthält weiterführende technische Informationen zur Realisierung der Hardware und Software der Multifunktions-Adapterplatine. Dieser Abschnitt enthält u.a. die Schaltpläne, Platinenlayouts und Bestückungslisten der einzelnen physisch realisierten Hardwarekomponenten. Einen Überblick über die Möglichkeiten zur Konfiguration der Multifunktions-Adapterplatine gibt die Übersicht der Konfigurationsbefehle.

## 2 Grundlagen zu den physischen Schnittstellen

### 2.1 Die klassischen seriellen Peripherieschnittstellen

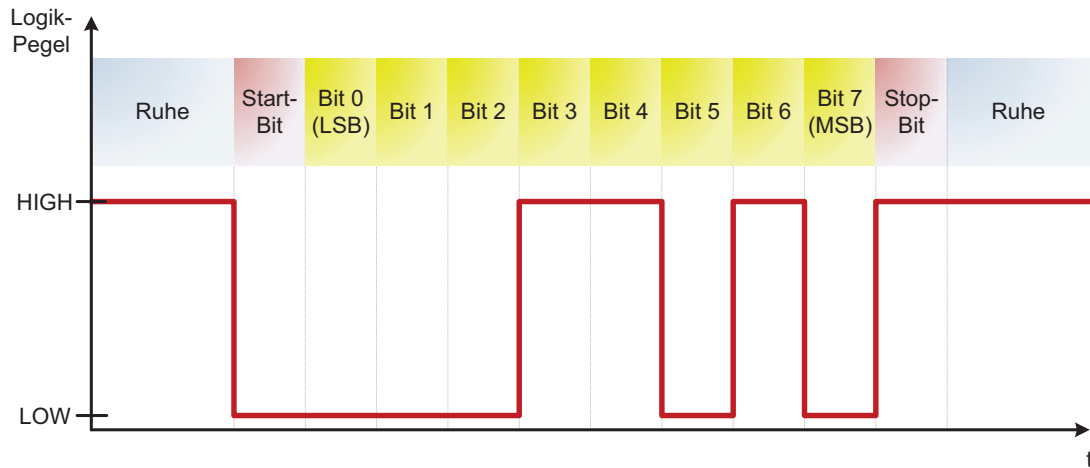
#### 2.1.1 Erzeugung des Datenpaketes auf Logikebene

Für die nachfolgend aufgeführten drei Schnittstellen erzeugt der UART die erforderliche Logik. Er dient der Realisierung einer digitalen seriellen Schnittstelle. Als elektrisches Bauelement, integriert in allen moderneren Mikrocontrollern, ermöglicht der UART die direkte Kommunikation zwischen diesen Komponenten auf nicht invertierenden Pegeln der Transistor-Transistor-Logik (TTL) bzw. Complementary Metal Oxide Semiconductor (CMOS).

Die Übertragung der Daten erfolgt dabei asynchron, d.h., es wird keine zusätzliche Leitung zur Übermittlung eines Taktsignales verwendet. Die zu übertragenden Daten, welche aus einem Wort von 5 bis 9 Bit Länge bestehen können, werden in einem Datenpaket übertragen und müssen vom Empfänger zu jedem beliebigen Zeitpunkt erkannt werden. Den Anfang eines Datenpaketes markiert deshalb stets das Start-Bit. Das Ende eines Datenpaketes signalisieren 1 bis 2 Stop-Bits. Durch den so gebildeten Rahmen aus Start- und Stop-Bits erfolgt die Synchronisierung zwischen Sender und Empfänger. Die als Nutzlast enthaltenen Daten werden mit dem Least Significant Bit (LSB) voran übertragen. Das Ende der Nutzlast markiert das Most Significant Bit (MSB). Damit der Empfänger zuverlässig die Daten des Senders auswerten kann, ist bei beiden Teilnehmern die identische Einstellung der Baudrate notwendig. Im Spezialfall dieser binären Modulation, es existiert hier nur der logische HIGH- oder LOW-Pegel, entspricht die Baudrate der Übertragungsrate in Bit pro Sekunde [23].

Der UART bietet darüber hinaus auch die Möglichkeit, dem Datenpaket ein Paritätsbit hinzuzufügen. Dadurch besteht die Möglichkeit, Bit-Fehler in ungerader Anzahl im Datenpaket festzustellen. Sender als auch Empfänger können hierzu auf Hinzufügen eines geraden oder ungeraden Paritätsbits eingestellt werden. Bei gerader Parität (*engl.*: even parity) ergibt die gerade Anzahl der logischen EINSEN in dem zu übertragendem Datenwort ein Paritäts-Bit vom Wert EINS. Bei Verwendung der ungeraden Parität (*engl.*: odd parity) wird die gerade Anzahl von EINSEN als logische NULL symbolisiert. Der Empfänger kann so ein fehlerhaft übertragenes Datenpaket verwerfen. Eine Fehlerlokalisierung bzw. Fehlerkorrektur ist damit jedoch nicht möglich.

Für den durch den UART erzeugten Logikpegel existiert keine Spezifikation, auf welchem Spannungspegel oder mit welcher Geschwindigkeit die Daten zu übertragen sind.



**Abbildung 2.1:** Erzeugter Logikpegel durch den UART in 8-N-1 Codierung

Lediglich der Logik-Pegel ist spezifiziert. Dies führte zur Einführung detaillierter spezifizierter Schnittstellenstandards, wie beispielsweise RS-232 und RS-422/485. Für das Erzeugen des entsprechenden Signalpegels werden spezielle Treiber-Bausteine verwendet.

Als Beispiel zur Sendung des Zeichens **X**, es entspricht dem dezimalen Wert 88 aus dem American Standard Code for Information Interchange (ASCII), dient hier die Abb. 2.1. Die Darstellung zeigt den Signalverlauf in der 8-N-1 Codierungsform. Diese gängige Schreibweise beschreibt das Datenpaket mit einer Wortlänge von 8 Bit, dem Verzicht auf ein Paritätsbit und der Verwendung von einem Stop-Bit.

Waren in der Vergangenheit die Übertragungsraten der mittels durch den UART erzeugten Signale auf wenige Bits pro Sekunde begrenzt, so erreichen sie heute mehrere MBit pro Sekunde. Die Voraussetzung für eine erfolgreiche Kommunikation zwischen Sender und Empfänger ist erfüllt, bei gleicher Konfiguration beider Teilnehmer nach folgenden Kriterien:

- Wortlänge
- Anzahl der Stop-Bits
- Übertragungsrate
- Parität

### 2.1.2 RS-232 Interface

Bereits in den 60er Jahren wurde der amerikanische Standard RS-232 von der Electronic Industries Association (EIA), später in Electronic Industries Alliance umbenannt, veröffentlicht. Die aktuelle Fassung der Norm lautet jedoch auf **TIA/EIA-232-F**. Diese Schnittstellennorm fand anfangs Verwendung zur Realisierung der Kommunikation zwischen Data Terminal Equipment (DTE), der Dateneneinrichtung, und dem Data Circuit-

Terminating Equipment (DCE), bei dem es sich meistens um ein Modem handelte. Der Datenverkehr erfolgt als reine Punkt-zu-Punkt Verbindung zwischen zwei Teilnehmern.

Die weite Verbreitung der RS-232 Schnittstelle resultiert aus der einfachen und preiswerten Realisierung der Kommunikation zwischen zwei Geräten. Für den Datenaustausch werden als Minimum nur drei Leitungen benötigt:

- Sendeleitung (TXD)
- Empfangsleitung (RXD)
- gemeinsame Bezugserde (GND)

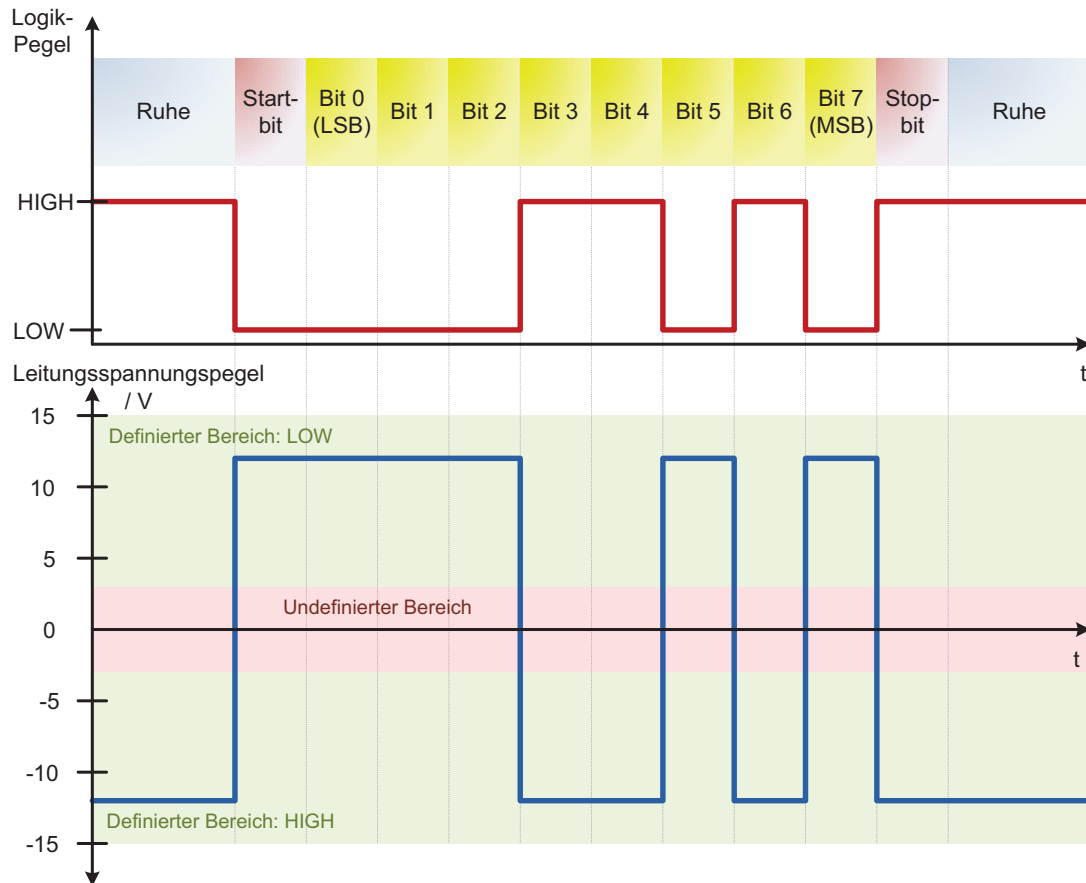
Durch die zwei entgegengesetzt gerichteten Datenleitungen entsteht so eine Voll-Duplex-Verbindung.

Resultierend aus der anfänglich vorgesehenen Verwendung dieser Norm zum Verbinden des Terminals mit dem Modem beschreibt die RS-232 Norm weitere separate Steuerleitungen, die der Ansteuerung des Modems dienen. Als Auswahl erfüllen diese Steuerleitungen die Funktionen, die im Standard V.24 [20] von der Comité Consultatif International Téléphonique et Télégraphique (CCITT), welche 1993 durch die International Telecommunication Union (ITU) abgelöst wurde [23], beschrieben sind.

Die elektrischen Eigenschaften entsprechen der Festlegung gemäß CCITT V.28. Der durch den UART erzeugte logische Pegel wird dabei durch entsprechende Treiberbausteine in ein invertiertes Signal umgewandelt. Aus der logischen NULL wird auf Leitungsebene ein Pegel von  $+3V$  bis  $+15V$  erzeugt, die logische EINS wird durch einen Pegel von  $-3V$  bis  $-15V$  symbolisiert. Leitungsspannungspegel im Bereich von  $-3V$  und  $+3V$  gelten als undefiniert. Die optionalen Steuersignale werden als nicht-invertierend ausgewertet und entsprechen dem Status aktiv, der logischen EINS, wenn der Spannungspegel jenseits der  $+3V$  liegt. Sinkt der Leitungsspannungspegel unterhalb der  $-3V$  Marke, so wird das Steuersignal als nicht aktiv, der logischen NULL, bewertet. Abbildung 2.2 zeigt die Umsetzung des logischen Pegels auf den Leitungsspannungspegel der TXD Datenleitung.

Die Leitungsspannungspegel sind auf einen Lastwiderstand  $R_{LOAD} = 3 - 7k\Omega$  bezogen. Diese hochohmige Übertragung führt zu einer hohen Störempfindlichkeit durch Einstreuung von Störungen durch andere Geräte. Die maximale Leitungslänge für die sichere Übertragung der Daten wird daher in der RS-232 Norm mit  $15m$  bei einer Übertragungsrate von  $19200Baud$  beziffert. Die Norm bezieht sich dabei auf eine maximalen Leitungskapazität von  $2,5nF$ , welche bei Standardkabeln einer Länge von  $15m$  entspricht. Durch Verwendung von Kabeln mit geringerer Leitungskapazität, wie zum Beispiel CAT-5 Kabel in der Ausführung Unshielded Twisted Pair (UTP), läßt sich die Übertragungsreichweite verdoppeln bis verdreifachen [23].

Die Normierung der RS-232 Schnittstelle legt als physikalische Schnittstelle zur Kontaktierung der Signalleitungen lediglich einen 25-poligen D-SUB Stecker fest. Die Norm gibt

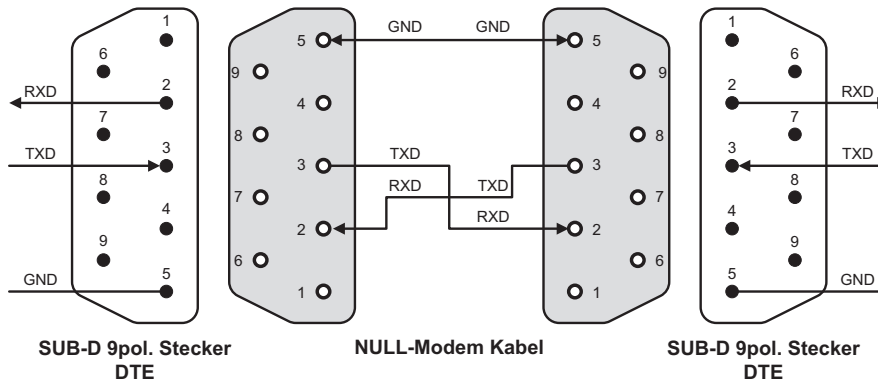


**Abbildung 2.2:** Umsetzung des UART-Pegels in RS-232 Spannungspegel

die eindeutige Kontaktierung aus Sicht des DTE zum DCE wieder. Da im 21sten Jahrhundert die Bedeutung der Verwendung von Modems zur Verbindung mehrerer Rechner im Netzwerk über die analoge Telefonleitung unbedeutend ist, wird in heutigen Ausführungen der RS-232 Schnittstelle auf die Durchkontaktierung der meisten Steuersignale verzichtet. In Computer- und Peripheriegeräten hat sich damit die Verwendung der 9-poligen D-SUB Steckverbindung als Quasi-Standard durchgesetzt. In normgerechten Ausführungen besitzt das als DTE bezeichnete Gerät einen Stecker (männlich, ausgeführter Anschluss mit Stiften) und das DCE eine Buchse (weiblich, passgenaues Gegenstück) [20]. Bei Anschluss eines DTE an ein DCE findet ein Kabel mit 1:1 Kontaktierung der erforderlichen Leitungen statt. Verbindet man dagegen zwei DTE miteinander, um zum Beispiel Daten zwischen Mikrocontroller und PC auszutauschen, so sind in der Minimalbelegung von drei Leitungen die TXD und RXD Leitung zu kreuzen. Man spricht hier von der Verwendung eines NULL-Modem Kabels (NULL, da ja kein Modem mit dem Gerät verbunden wird). Die Abbildung 2.3 stellt die physische Verbindung von zwei DTE-Geräten dar.

### 2.1.3 RS-422 Interface

Als Weiterentwicklung der RS-232 Schnittstelle stellt die RS-422 Schnittstelle eine deutliche Verbesserung der Übertragungseigenschaften dar. Ihre korrekte Bezeichnung lautet



**Abbildung 2.3:** Verbindung zweier DTE über 9pol. D-SUB Verbindung

in der aktuellen Fassung der Norm **TIA-422-B**. Durch den Einsatz einer symmetrischen Signalübertragung wird eine deutlich höhere Übertragungsgeschwindigkeit bei gesteigerter Übertragungsqualität erreicht. Dadurch ist es möglich, einen deutlich längeren Übertragungsweg sicher zu überbrücken.

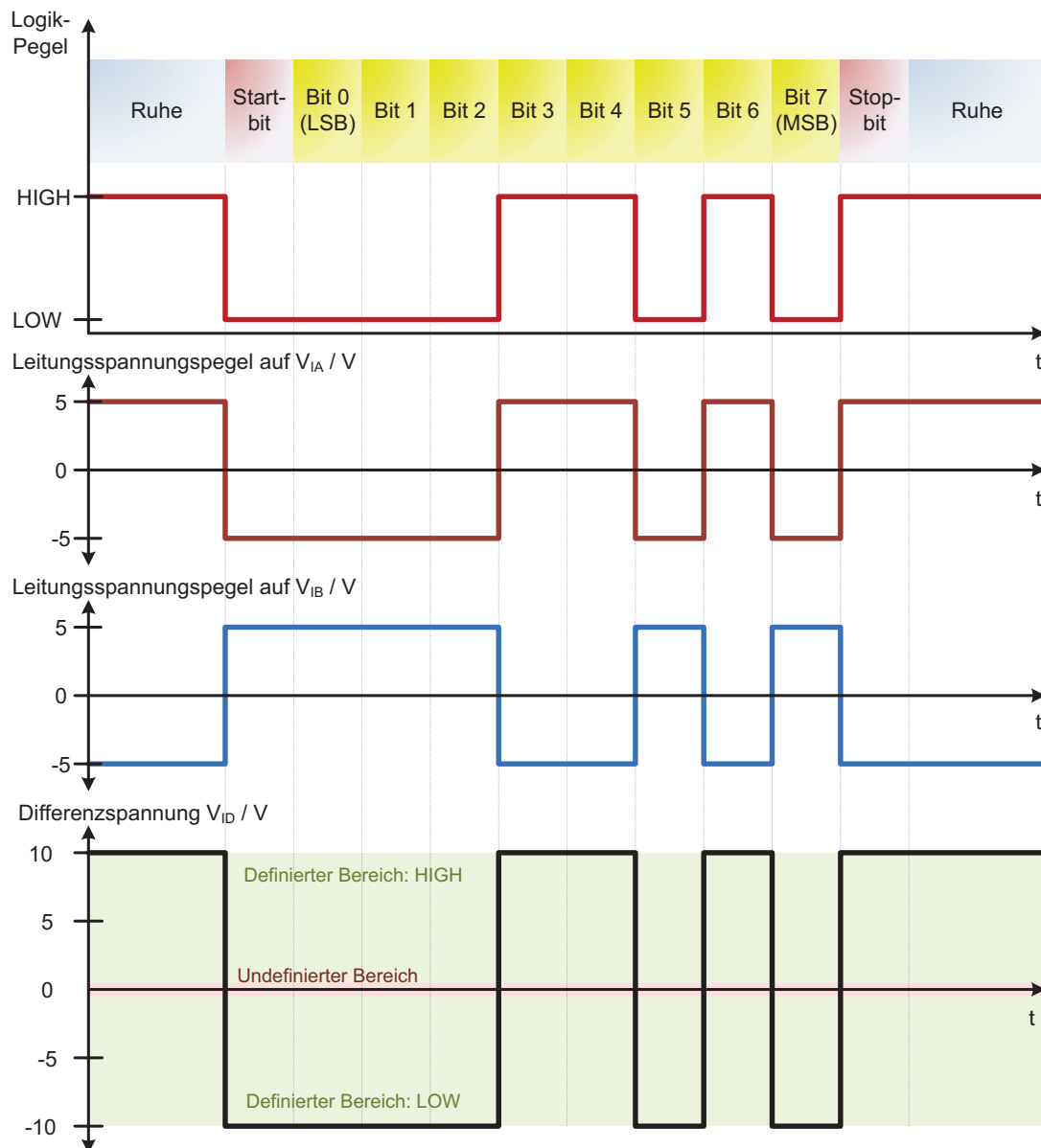
Für die Übertragung der Signale in jeweils einer Richtung werden zwei Signalleitungen benötigt. Auf der einen Leitung wird das nicht-invertierte Signal des Treibers, der hier als Sender agiert, übertragen. Auf der zweiten Datenleitung wird das komplementäre invertierte Signal übertragen. Die Norm enthält dabei keine explizite Bezeichnung der einzelnen Datenleitung. Aber im Allgemeinen wird aus Sicht des Empfängers die nicht-invertierende Signalleitung als **A** bezeichnet. Die invertierte Leitung erhält hier die Bezeichnung **B**. Im Simplex Betrieb gilt diese Bezeichnung der Datenleitungen auch für den Sender. In einer Voll-Duplex Struktur existieren insgesamt vier Datenleitungen. Die nicht-invertierte Datenleitung des Senders erhält dann die Bezeichnung **Y**, während die invertierte Leitung als **Z** bezeichnet wird. Der vom Sender erzeugte Spannungspegel auf jeder Datenleitung sollte entsprechend der Spezifikation im Bereich von  $\pm 6V$  liegen. Für die am Empfänger anliegende Gleichtaktspannung (engl.: *common mode voltage*) wird ein Wert im Bereich von  $V_{CM} = \pm 7V$  toleriert [10]. Die Gleichtaktspannung  $V_{CM}$  ist dabei der Mittelwert der am Eingang des Empfängers zu einem gemeinsamen Zeitpunkt anliegenden Spannungspegel gemäß der Formel 2.1.

$$V_{CM} = \frac{V_{IA} + V_{IB}}{2} \quad (2.1)$$

Durch den Einsatz eines Differenzverstärkers mit nachgeschaltetem Komparator im Empfängerbaustein kann so zuverlässig der **HIGH**- bzw. **LOW**-Pegel des Datensignals bestimmt werden. Gilt für die Differenz der Spannungspegel der Leiter **A** und **B**:  $V_{ID} = V_{IA} - V_{IB} \geq 200mV$ , dann erhält das Empfangssignal den Zustand der logischen **EINS**. Ergibt die Differenz beider Datenleitungen einen Wert nach  $V_{ID} = V_{IA} - V_{IB} \leq -200mV$ , so wird eine logische **NULL** interpretiert. Der Bereich zwischen  $\pm 200mV$  gilt als undefiniert. Gemäß der Norm sollte der Differenzspannungspegel  $V_{ID}$  im Bereich von  $\pm 10V$  liegen.

Die Spezifikation der RS-422 Schnittstelle sieht vor, dass ein aktiver Sender bis zu 10

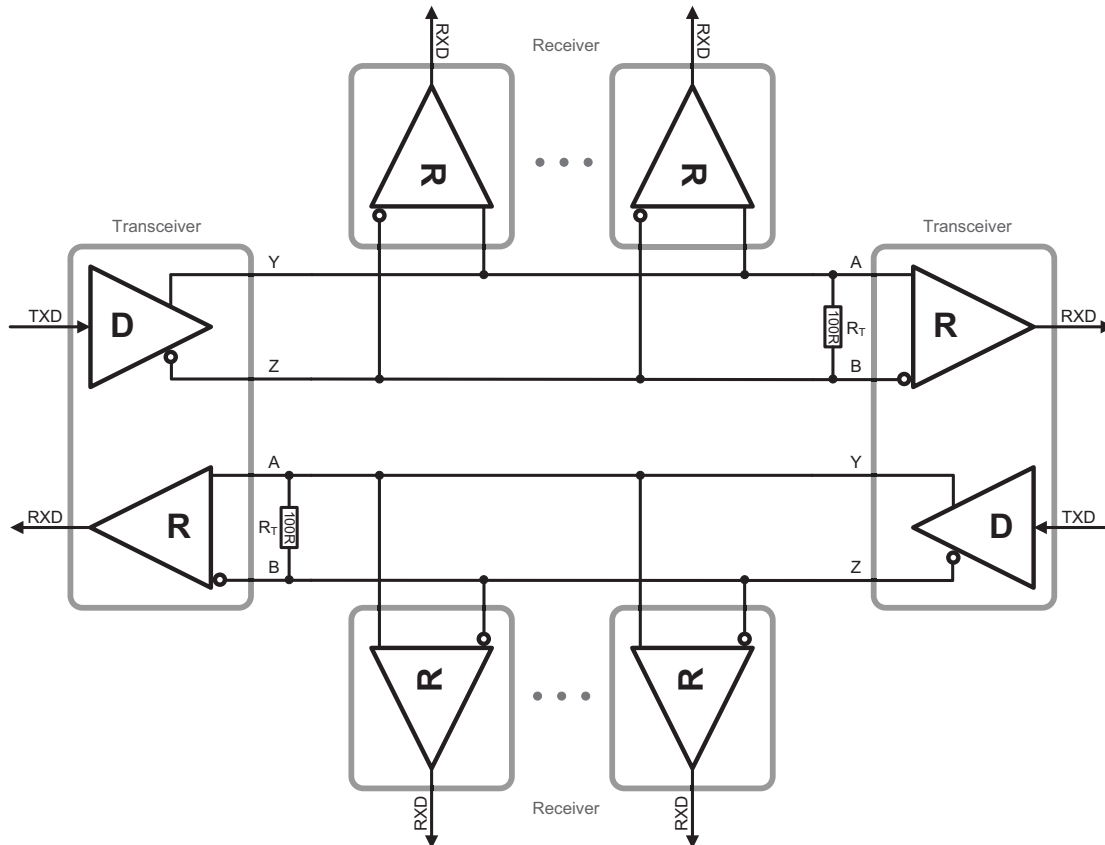
Empfänger bedient. Damit geht diese Schnittstelle gegenüber der RS-232 Schnittstelle den Schritt weiter von einer reinen Punkt-zu-Punkt Verbindung hinüber zu einem Bussystem, welches eine Vielzahl von Teilnehmern bedient. Als Eingangswiderstand wird hier ein Wert jedes Empfängers von mindestens  $R_{LOAD} = 4k\Omega$  gefordert. Ein weiterer Punkt der Spezifikation legt fest, dass nur ein Sender pro Busstrang vorhanden sein darf [22]. Ein nach der RS-422 Norm gebildetes Bussystem wird als *Simplex Multidrop* bezeichnet. Durch Hinzufügen eines zweiten Busstranges in umgekehrter Übertragungsrichtung wird eine Voll-Duplex Verbindung ermöglicht. Beispielhaft soll Abbildung 2.5 diese Busstruktur verdeutlichen.



**Abbildung 2.4:** Symmetrische Signalübertragung und Differenzsignalbildung

Zur Übermittlung der Daten ist als Medium ein verdrehtes Kabelpaar vorgesehen. Die in solchen Kabeln eingestreuten Störimpulse treten dann in beiden Leitern gleichzeitig auf

und werden durch die Differenzbildung des Empfangssignales nahezu vollständig eliminiert. Um Signalreflexionen am Ende der Datenleitung zu unterdrücken, lässt sich durch den Einsatz eines Terminierungswiderstandes die Übertragungsreichweite und/oder die Übertragungsgeschwindigkeit erhöhen. Der Terminierungswiderstand wird üblicherweise mit  $R_T = 100\Omega$  und wird dem Empfänger direkt zwischen den beiden Datenleitungen A und B vorgeschaltet [21].



**Abbildung 2.5:** RS-422 Bussystem in Full-Duplex Konfiguration

Bei Abschalten des Senders, Senderausfall oder einer Störung im Übertragungskanal (z.Bsp. Kabelbruch) kommt es durch die Differenzbildung zu einem undefinierten bzw. fehlerhaft interpretierten Spannungspegel. Abhilfe schafft hier die Verwendung der Fail-Safe Schaltung. Sie sorgt für einen definierten Spannungspegel auf den Datenleitungen, der durch die Empfänger als Ruhe-Zustand interpretiert wird. Dazu wird das Potential der nicht-invertierenden Datenleitung A durch Einsatz eines Pull-Up Widerstandes auf ein positives Spannungsniveau angehoben. Die invertierende Datenleitung B wird mittels Pull-Down Widerstand auf ein Massepotential gezogen. In den modernen Sender-/Empfänger-Bausteinen ist diese Schaltung bereits integriert [10].

Durch die Verwendung der symmetrischen Signalübertragung werden gegenüber der unsymmetrischen Übertragung durch die RS-232 Schnittstelle deutliche Verbesserungen der Übertragungsgeschwindigkeit bei gesteigerter Übertragungsreichweite erzielt. So lassen sich zuverlässig über eine kurze Distanz von ca. 10m bis zu 10MBit/s

übertragen. Bei Nutzung der maximal normierten Leitungslänge von  $1200m$  wird immerhin noch eine Datenübertragungsrate von  $90kBit/s$  erreicht. Die erreichbare Datenübertragungsrate bei entsprechender Leitungsdistanz hängt dabei maßgeblich von dem verwendeten Übertragungsmedium und der entsprechenden Wahl der Terminierung ab.

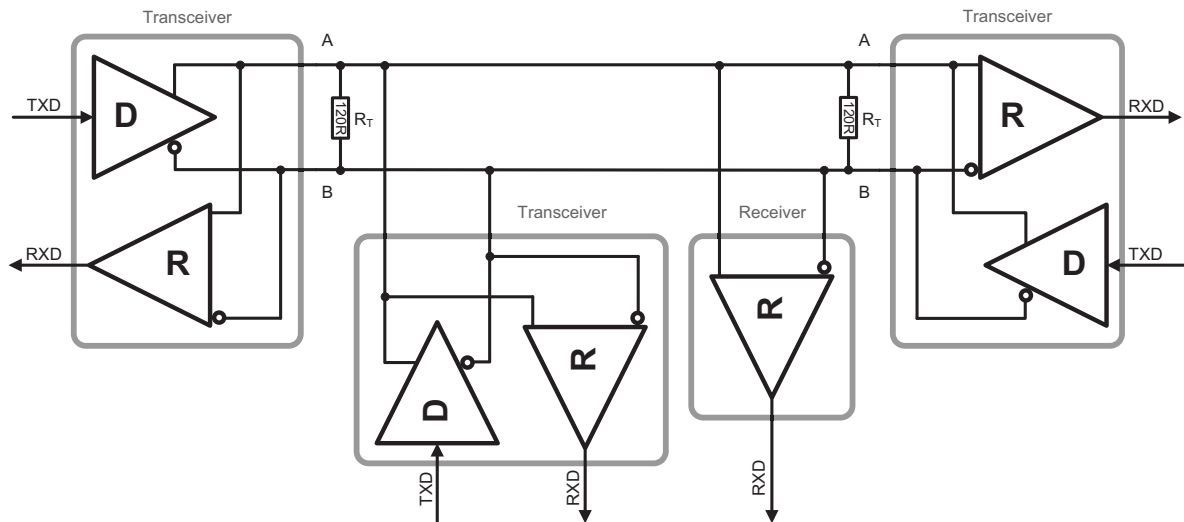
#### 2.1.4 RS-485 Interface

Die RS-485 Schnittstelle, in der aktuellen Norm als **TIA-485-A** bezeichnet, stellt eine Weiterentwicklung der RS-422 Schnittstelle dar. Auch wenn diese beiden Schnittstellentypen viele Ähnlichkeiten aufweisen, so existieren doch diverse Unterschiede in der Spezifikation und dem daraus resultierenden Anwendungsgebiet.

Beiden Schnittstellen gemeinsam ist die symmetrische Übertragung der Daten auf zwei Leitungen pro Übertragungsrichtung. Die maximal am Empfänger anliegende Gleichtaktspannung  $V_{CM}$  beträgt  $-7V$  bis  $+12V$ . Die maximal zulässige Anzahl der am Bussystem beteiligten Empfänger beträgt 32. Voraussetzung dafür ist, dass jeder Empfänger einen Lastwiderstand von  $R_{LOAD} = 12k\Omega$  entspricht. Dies führt zur Definition des Unit Load (UL). Ein UL entspricht demnach einem Empfänger mit diesem Eingangswiderstand. Der RS-485 Norm konform ist es aber ebenso, Empfänger mit beispielsweise  $0.5 * UL$  einzusetzen. Ein derartiger Empfänger besitzt dann einen doppelt so hohen Eingangswiderstand von  $R_{LOAD} = 24k\Omega$ . Durch Verwendung nur derartiger Empfänger ist der Treiber-Baustein (Sender) in der Lage, 64 Empfänger zu bedienen [3]. Der Einsatz von Empfängern mit noch höheren Eingangswiderstand führt zur weiteren Verringerung des UL, wodurch der Einsatz von noch mehr Empfängern innerhalb des Bussystems möglich wird.

Durch entsprechende Steuerleitungen oder Logik-Ansteuerung empfiehlt sich die RS-485 Schnittstelle zum Aufbau von Halb-/Voll-Duplex Bussystemen in Multipoint Struktur. Bedingung ist allerdings, dass zu einem Zeitpunkt nur ein Sender aktiv sein darf. Abbildung 2.6 zeigt als Beispiel den Aufbau eines RS-485 Bussystems in Halb-Duplex Konfiguration.

Arbeiten in einem solchen Bussystem mehrere Sender gleichzeitig, so tritt der Fall der Sender-Konkurrenz ein. Dieser auftretende Fehlerstrom führt in jedem aktiven Sender zu einer hohen Verlustleistung. Daher müssen in den Senderbausteinen entsprechende Schutzmaßnahmen getroffen werden, die eine dauerhafte Beschädigung verhindern. Dies geschieht zum Einen durch die Verwendung einer Strombegrenzung im Sender. Dadurch kann der Sender nach sofortiger Beendigung der Sender-Konkurrenz wieder seine Arbeit aufnehmen. Eine zweite Maßnahme ist die Integration eines thermischen Überlastungsschutzes. Dieser verfügt zwar über eine relativ lange Wiedereinschaltzeit des Senders, ermöglicht aber auch, neben dem Erkennen eines zu hohen Stromes, die Detektion einer thermischen Überlastung [20].



**Abbildung 2.6:** RS-485 Bussystem in Halb-Duplex Konfiguration

Um Reflexionen auf dem Übertragungsmedium zu vermeiden, kommen im einfachsten Fall Terminierungswiderstände zum Einsatz. In der Halb-Duplex Konfiguration des RS-485 Bussystems werden Widerstände jeweils am äußeren Ende des Busstranges zwischen den Datenleitungen A und B platziert. Der Terminierungswiderstand  $R_T$  ist dabei so zu dimensionieren, dass er genau dem Wellenwiderstand  $Z_0$  des Kabels entspricht [2]. Eine gängige Dimension des Terminierungswiderstandes ist mit  $R_T = 120\Omega$  gegeben.

Die Interpretierung des Signals durch den Empfänger durch Bildung des Differenzsignals geschieht, identisch zum RS-422 Standard, mit identischen Spannungspegeln. Auch die erzielbare Datenübertragungsrate bei entsprechender Überreichweite ist bei beiden Schnittstellennormen identisch. Durch den erweiterten Bereich der Gleichtaktspannung lassen sich RS-485 Empfänger problemlos in bestehende RS-422 Bussysteme integrieren. In umgekehrter Integrationsweise ergibt sich hier eine Inkompatibilität hinsichtlich der vertretbaren Gleichtaktspannung, die geringe Last führt zu einer hohen UL und das eventuelle Fehlen von Schutzmaßnahmen bei Auftreten von Sender-Konkurrenz.

## 2.2 Nahbereichskommunikation auf Platinenebene

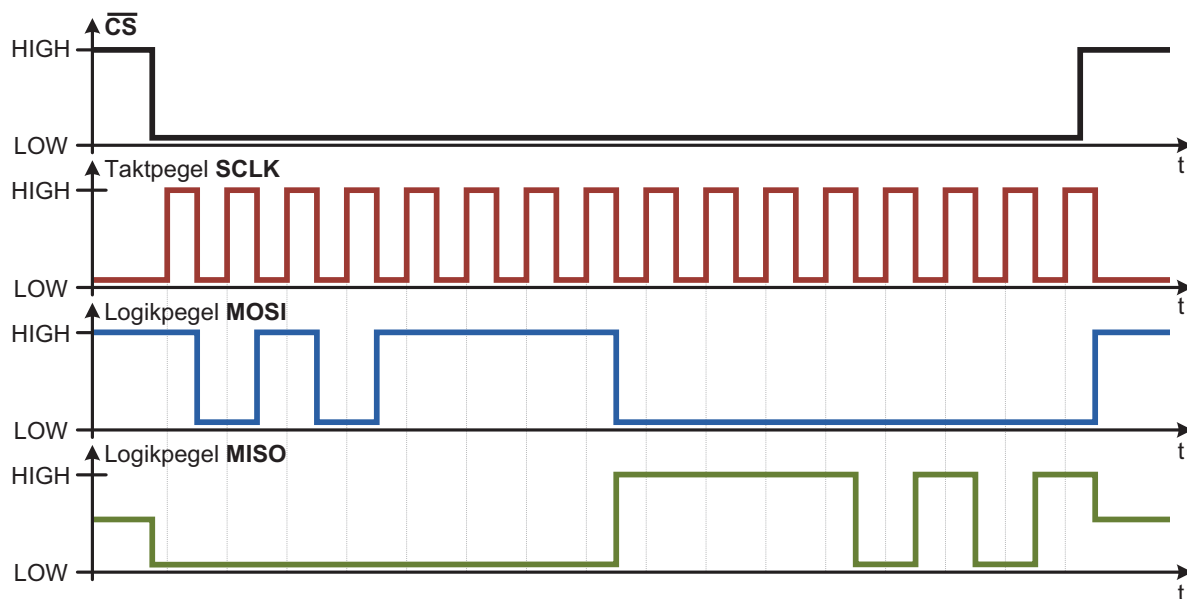
### 2.2.1 Serial Peripheral Interface

Das von Motorola entwickelte SPI dient der Kommunikation zwischen Integrated Circuits (IC) auf Platinenebene. Durch die serielle synchrone Datenübertragung in beiden Richtungen erfolgt der Datenaustausch zwischen einem Master und einem bis mehreren Slaves.

Im 3-Wire Modus dieser Schnittstellennorm werden drei Leitungen für eine bidirektionale Kommunikation benötigt:

- SPI Clock (SCLK), dient der synchronen Datenübertragung
- Master Out Slave In (MOSI), die Sendeleitung aus Sicht des Masters
- Master In Slave Out (MISO), die Empfangsleitung aus Sicht des Masters

Zur Verbindungsaufnahme wird noch eine weitere Steuerleitung zur Auswahl des Slaves benötigt. Diese ist **LOW**-Aktiv und wird meist als CS (Chip Select) bezeichnet. Die fallende Flanke des CS Signals teilt dem Slave den Beginn einer Kommunikation mit dem Master mit. Auf der MOSI Leitung übersendet der Master dem Slave dann seine Daten und der Slave antwortet im selben Zeitfenster über die MISO Leitung. Der Datenverkehr erfolgt synchron im Takt des SCLK. Benötigt der Slave keine Daten vom Master, um seine Aufgabe abzuarbeiten, kann die MOSI Leitung eingespart werden. Hier handelt es sich um den 2-Wire Modus der SPI Spezifikation. Die steigende Flanke des CS Signals beendet die Kommunikation und versetzt den Slave wieder in den Ruhezustand [7]. Abbildung 2.7 zeigt hier exemplarisch die Master-Slave Kommunikation, wie sie zum Beispiel zwischen MCU und ADC stattfinden könnte.



**Abbildung 2.7:** Beispiel einer 16-Bit SPI-Kommunikation im Modus 0

Der SPI Standard definiert kein spezielles Protokoll nachdem die Datenübertragung zu erfolgen hat. Jedoch existieren zwei Parameter, die in ihrer Kombination zur Bildung von vier Übertragungsmodi führen. Die Clock Polarity (CPOL) gibt an, auf welchem Logik-Pegel das SCLK Signal im Ruhezustand ist. Mit CPOL=0 befindet sich das SCLK-Signal im Ruhezustand auf dem Pegel LOW. Nach dem Setzen des CS-Signals auf den Zustand LOW ist die erste Flanke des SCLK-Signals eine Steigende, gefolgt von der zweite Flanke, die eine Fallende ist. Mit der Konfiguration von CPOL=1 verhält sich dieser Sachverhalt genau umgekehrt. Dann ist die erste Flanke des SCLK-Signals eine fallende Flanke, die Zweite ist eine steigende Flanke. Die Clock Phase (CPHA) bestimmt, bei

welcher Flanke des SCLK-Signals das Datenbit übernommen wird. Mit CPHAS=0 erfolgt die Datenübernahme zum Zeitpunkt der ersten Flanke des SCLK-Signals, bei CPHAS=1 erst bei Auftreten der zweiten Flanke. Tabelle 2.1 enthält die zum Übertragungsmodus entsprechende Konfiguration der beiden Parameter.

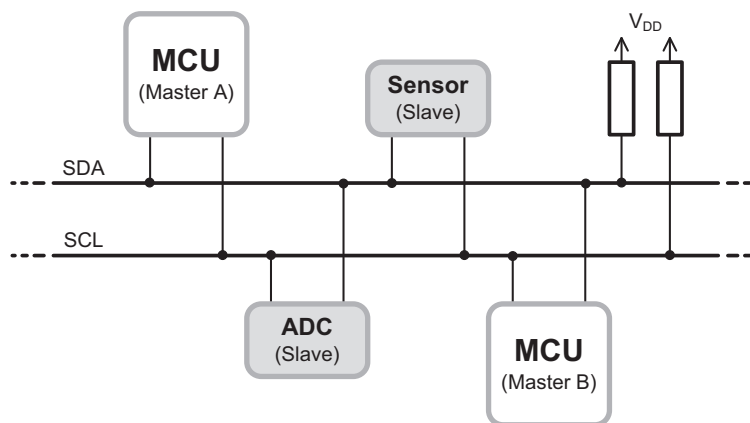
| Modus | CPOL | CPHA |
|-------|------|------|
| 0     | 0    | 0    |
| 1     | 0    | 1    |
| 2     | 1    | 0    |
| 3     | 1    | 1    |

**Tabelle 2.1:** SPI-Übertragungsmodi

Die SPI Spezifikation enthält keine Angaben, wie lang das zu übermittelnde Datenwort zu sein hat. Ebenso gibt es zum SCLK keine Vorgaben zur verwendeten Frequenz. Diese ist frei wählbar und reicht von ein paar kHz bis hin zu einigen MHz. Durch diese Freiheitsgrade ist SPI weit verbreitet und dient heute zur Anbindung von ADCs, Speicher, Sensoren und Controllern an die MCU. Durch die Vielfalt der bestehenden SPI Implementierungen ist allerdings eine zweckgebundene Anpassung der Software nach Studium des Datenblattes des entsprechendem Peripheriebausteins fast immer erforderlich.

## 2.2.2 Inter Integrated Curcuit Bus

Der bereits in den frühen 80er Jahren von Philips entwickelte  $I^2C$  dient der Kommunikation von ICs untereinander bei geringen bis mittleren Übertragungsraten. Reduziert auf zwei Leitungen erfolgt bei diesem Bus die Master-Slave Kommunikation durch bidirektionalen seriellen Datenaustausch. Der  $I^2C$ -Bus erlaubt dabei das Vorhandensein von mehreren Mastern im Bussystem und bildet so ein echtes Multi-Master Bussystem. Die Spezifikation sieht dazu eine Kollisionserkennung und eine Bus-Arbitrierung vor. Eine beispielhafte Buskonfiguration kann der Abbildung 2.8 entnommen werden.



**Abbildung 2.8:**  $I^2C$ -Bus mit mehreren Mastern

Alle Master- und Slaveeinheiten kommunizieren bei dem I<sup>2</sup>C-Bus über die beiden Signalleitungen Serial Clock Line (SCL) und Serial Data Line (SDA). Die beiden Signalleitungen werden über Pull-Up Widerstände mit der Versorgungsspannung  $V_{DD}$  verbunden. Der Spannungspegel der Signale ist so nicht eindeutig definiert und richtet sich nach der anliegenden Versorgungsspannung, um eine große Bandbreite von IC-Technologien zu unterstützen. Für CMOS-ICs gilt hier, dass ein HIGH-Pegel interpretiert wird, wenn  $V_{IN} \geq 0,7 * V_{DD}$  ist. Der Pegel gilt als LOW, wenn  $V_{IN} \leq 0,3 * V_{DD}$  beträgt[12].

Die Kommunikation über den I<sup>2</sup>C-Bus geht immer vom Master aus. Zu Beginn einer Sendung wird eine Start Kondition auf der Leitung des Bussystems erzeugt, gefolgt von dem 8-Bit langen Adresswort. Dieses enthält in den ersten sieben Bit die Adresse des Slaves, das achte Bit gibt die Richtung der Datenverbindung (NULL für Senden, EINS für Empfangen) an. Abzüglich der reservierten Adressen lassen sich so 112 Teilnehmer am Bus adressieren. Durch Erweiterung des Adressbereiches auf 10-Bit in jüngeren Spezifikationen lässt sich die Anzahl der Teilnehmer noch deutlich steigern. Nach Senden des Daten-/Adressbytes folgt als neuntes Bit das Acknowledge (ACK) bzw. Not Acknowledge (NACK). Für diese Taktphase gibt der Master den Bus frei und der Slave sendet entsprechend seinem Verständnis das ACK oder NACK. Dem Master obliegt es danach, die Übertragung fortzusetzen, durch Generierung einer Stop Kondition die Kommunikation abubrechen, oder eine weitere Start Kondition zu erzeugen, um die Übermittlung zu wiederholen[6].

Die vom Master auf der SCL Leitung angelegte Frequenz muss im Maximum den vier vorgesehenen Übertragungsmodi entsprechen. Die Übertragungsmodi können der Tabelle 2.2 entnommen werden. Der angesprochene Slave hat dabei die Möglichkeit, die Übertragung der Datenbytes durch Clock Stretching zu verlangsamen, indem das SCL Signal vom Slave auf den Zustand LOW gehalten wird.

| Modus           | Maximale Übertragungsrate |
|-----------------|---------------------------|
| Standard-mode   | 100 kbit/s                |
| Fast-mode       | 400 kbit/s                |
| Fast-mode Plus  | 1 Mbit/s                  |
| High-speed mode | 3.4 Mbit/s                |

**Tabelle 2.2:** I<sup>2</sup>C-Übertragungsmodi

Kommen mehr als ein Master in diesem Bussystem zur Anwendung, so gilt es zu entscheiden, wem bei gleichzeitigem Senden der Zugriff auf den Bus zugesichert wird. Dazu ist jeder Empfänger mittels Open-Collector bzw. Open-Drain Schaltung mit dem Bus verbunden. Befindet sich nun ein Master im Sendemodus, so vergleicht er bei jedem HIGH-Pegel des SCL Signals, ob der gesendete Pegel auf SDA mit dem Soll identisch ist. Entspricht der SDA-Pegel LOW, während der Master ein HIGH gesendet hat, so liegt eine Sender-Konkurrenz vor und er verliert den Zugang zum Bus. Der betroffene Master schaltet dann unverzüglich seinen SDA-Treiber ab und überlässt dem verbleibenden Master den Buszugriff [6].

Der I<sup>2</sup>C-Bus eignet sich durch seine preiswerte und einfache Realisierung zur Anbindung vieler Peripheriebausteine an eine einzige MCU in einem aus nur zwei Leitungen bestehendem Bussystem. Überall dort, wo es nicht zwangsläufig um eine hohe Datenübertragungsrate ankommt findet dieser Bus Anwendung. So ist der I<sup>2</sup>C-Bus weit verbreitet bei der Verwendung von ADCs, Echtzeituhren, Sensoren und Flash-Speicher.

## 2.3 Universal Serial Bus

Die USB-Schnittstelle wurde erstmals 1996 von der Firma Intel eingeführt. Anfangs als Schnittstelle zum Anschluss weniger Geräte wie Maus, Tastatur und Drucker an den Personal Computer (PC) geplant, dient er heute durch ständige Weiterentwicklung der Spezifikation der Verbindung einer Vielzahl von Peripheriegeräten. Durch die Hot Plug and Play Fähigkeit, dem Entfernen und Hinzufügen von Teilnehmern an das Bussystem im laufenden Betrieb, hat der USB weitestgehend die klassische serielle Schnittstelle RS-232 und die parallele Schnittstelle aus dem PC-Sektor verdrängt.

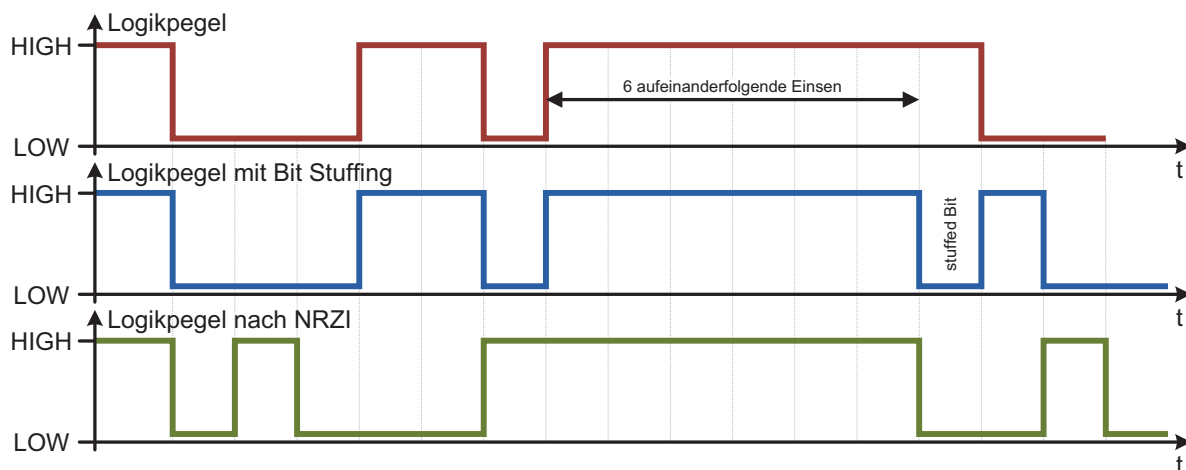
Die Übertragung der Daten erfolgt durch symmetrische Signale. Auf der mit **D+** bezeichneten Leitung wird das nicht-invertierte Signal, auf der Datenleitung **D-** das invertierte Signal übertragen. Diese Datenleitungen werden an beiden Enden durch jeweils einen Terminierungswiderstand  $R_T = 45\Omega$  gegen Masse abgeschlossen [1], welches der Terminierung des Wellenwiderstandes des Kabels von  $Z_0 = 90\Omega$  entspricht. Die USB Spezifikation sieht neben diesen beiden Datenleitungen auch das Mitführen je einer Leitung für Gleichspannung +5 V, diese Leitung wird als  $V_{BUS}$  bezeichnet, und Masse vor. Sie dienen der Spannungsversorgung von Peripheriegeräten ohne eigene Versorgung. Die Last ist dabei auf maximal 500 mA limitiert.

In einem USB System existiert genau ein Master, hier Host genannt. Dem Master folgt meist ein Hub, um weitere Peripheriegeräte mit dem Bus verbunden werden können. Ein Hub dient dabei nicht nur als Verteiler, er übernimmt auch die Aufgabe der Signalverstärkung. Ein USB System bildet so eine Baumstruktur aus an deren Spitze der Host steht. Dieser übernimmt die Konfiguration des Bussystems. Durch den Host erfolgt zentral das Erkennen der aktiven Busteilnehmer, die Vergabe einer eindeutigen Adresse an die Slaves und das Feststellen des Bandbreitenbedarfes eines jeden Einzelnen von Ihnen. Die Vergabe der Adressen an die Slaves erfolgt innerhalb eines 7-Bit breiten Adressraumes, womit maximal 127 Teilnehmer innerhalb eines Bussystems vorhanden sein können.

In der USB Spezifikation 2.0 werden Peripheriegeräte in drei unterschiedliche Geschwindigkeitsklassen eingeteilt. Diese Klassen sind:

- Low-Speed Mode mit 1,5 Mbit/s
- Full-Speed Mode mit 12 Mbit/s
- High-Speed Mode mit 480 Mbit/s

In den übermittelten Datenpaketen, die durch einen 16 Bit Cyclic Redundancy Check (CRC) verifiziert werden, sind die Daten als Payload mit dem LSB voran abgelegt. Zur Leitungscodierung wird hier auf das Verfahren des Non Return to Zero Invert (NRZI) zurückgegriffen. Der aktuell zu erzeugende Ausgangspegel hängt dabei vom Zustand des vorangegangenen Ausgangspegel ab. Für das erste Auftreten eines LOW im Logikpegel wird so auch der Leitungspegel zum LOW codiert. Das nächste Auftreten des LOW im Logikpegel, egal ob direkt aufeinanderfolgend oder erst einige Takte später im Datenpaket, wird als HIGH im Leitungspegel dargestellt. Die Invertierung der NULL wird in jedem Takt durchgeführt. Die Invertierung des logischen HIGH-Pegels erfolgt ebenso abhängig vom vorangegangenen Leitungspegel. Nur behalten aufeinanderfolgende logische EINSen in der Nachricht ihren aktuellen Leitungspegel bei. Um einen Leitungspegelwechsel im Falle der Aufeinanderfolge vieler HIGH-Zustände zu erzwingen, kommt hier das Verfahren des Bit Stuffing zum Einsatz. Nach sechs aufeinanderfolgenden EINSen im Logikpegel wird hierzu eine NULL eingefügt. Dies führt bei der Erzeugung des Leitungspegels automatisch zu einer Pegeländerung der aufeinanderfolgenden HIGH-Zustände [1]. Die schnellen Leitungspegeländerungen dienen dabei der Rückgewinnung des Taktsignals und somit der Synchronisierung des Datenstromes. Abbildung 2.9 soll beispielhaft die Leitungscodierung mittels NRZI verdeutlichen.



**Abbildung 2.9:** Leitungscodierung mit Bit Stuffing und NRZI

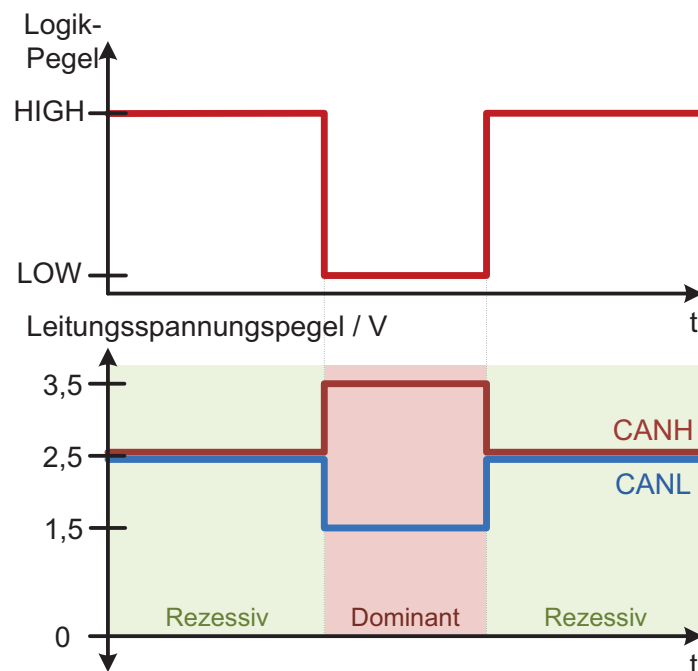
In der aktuellen Fassung der USB Spezifikation der Version 3.0 wurde eine Vielzahl von Veränderungen eingebracht. Im Wesentlichen wird hier die zulässige Leistungsaufnahme von Peripheriegeräten ohne eigene Spannungsversorgung auf 900 mA erhöht. Dies wurde unter anderem deshalb notwendig, da die Anlaufströme von externen Festplatten im 2.5 Zoll Format mit 600 mA bis 1100 mA deutlich über der Spezifikation der Version 2.0 liegen. Außerdem wird in der aktuellen Spezifikation eine neue Geschwindigkeitsklasse, namentlich als SuperSpeed bezeichnet, definiert, die eine Übertragungsrate von 4000 Mbit/s ermöglicht. Diese Neuerungen führen jedoch dazu, dass zur Nutzung dieser Norm neue Steckverbindungen und andere Kabelleitungen verwendet werden müssen.

## 2.4 Controller Area Network

Der CAN-Bus wurde von der Firma Robert Bosch GmbH entwickelt. Als Vertreter der Feldbussysteme verbindet es Sensoren und Aktoren miteinander. Ursprünglich konzipiert für den Einsatz in Kraftfahrzeugen dient es als Multi-Master Bussystem zur Übermittlung kleiner Datenmengen. Das CAN-Kommunikationsprotokoll ist durch die Norm ISO-11898 spezifiziert [11].

Ähnlich der RS-485 Schnittstelle erfolgt auch bei CAN die Kommunikation über zwei Leitungen durch ein symmetrisches Signal. Die Enden des Bussystems werden auch hier mit jeweils einem parallelem Terminierungswiderstand, der für gewöhnlich mit  $R_T = 120\Omega$  dimensioniert wird, abgeschlossen.

Der Leitungspegel wird als invertiertes Signal zum Logikpegel erzeugt und über die Leitung **CANH** übertragen. Die Leitung **CANL** enthält das dazu horizontal gespiegelte Signal. Dadurch entstehen die beiden logischen Zustände **Rezessiv**, für die Darstellung der logischen **EINS**, und **Dominant** für die logische **NULL**. Der Zustand **Rezessiv** liegt vor, wenn der Spannungspegel  $V_{CANH}$  kleiner als der Spannungspegel von  $V_{CANL} + 0.5V$  ist. Der Zustand **Dominant** wird erreicht, wenn die Spannung von  $V_{CANH}$  größer als  $V_{CANL} + 0.9V$  ist [9]. Abbildung 2.10 stellt die Erzeugung des Spannungspegels exemplarisch dar.



**Abbildung 2.10:** Wandlung des Logik-Pegels zum CAN-Leitungsspannungspegel

Das Datenpaket in einem CAN-Bussystem enthält bis zu 8 Byte an Daten. Diesem voraus geht der Identifier, der in der Standard-Spezifikation 11 Bit umfasst und in der erweiterten CAN-Spezifikation insgesamt 29 Bit enthält. Damit können den 30 maximal am Bus angeschlossenen Teilnehmern  $2^{11}$  bzw.  $2^{29}$  Kennungen zugewiesen werden. Der binäre

Wert des Identifiers bestimmt auch die Arbitrierung des Busses. Um Kollisionen zu vermeiden wird das gleiche Verfahren wie bei der  $I^2C$  Schnittstelle angewandt. Während ein Sender sein Datenpaket versendet tastet sein hochohmig geschalteter Empfänger den aktuellen Leitungszustand ab. Da der Zustand **Dominant** den Zustand **Rezessiv** überschreibt, verliert der Sender mit dem Ausgangspegel **Rezessiv** die Arbitrierung und bricht die Sendung umgehend ab. Da der Identifier der erste Teil des CAN-Datenpaketes ist, entscheidet sich bereits hier, welcher Teilnehmer Zugriff zum Bus erhält. Damit lassen sich die Kennungen priorisieren. Ein hoher binärer Wert des Identifiers entspricht demnach einer niedrigen Priorität. Einem niedrigen binären Wert des Identifier wird eine hohe Priorität eingeräumt.

Die erreichbare Datenübertragungsrate ist auch beim CAN-Bus von der Länge der Übertragungsleitung und der Art der Terminierung abhängig. Als obere Grenze für die Datenübertragungsrate sind  $1\text{Mbit/s}$  spezifiziert, bei der sich ca. 40 m Signalleitung überwinden lassen. Bei einer Länge der Übertragungsleitung von 2 km ist allerdings eine Übertragungsrate von nur noch ca.  $20\text{kbit/s}$  möglich [23].

## 2.5 Ethernet Interface

Das Ethernet spezifiziert Protokolle (Software) und Übertragungsmedien (Hardware) für kabelgebundene Netzwerke. Seine ursprünglich zugesprochene Aufgabe ist das Verbinden von Teilnehmern innerhalb eines Local Area Network (LAN), welches in Heimnetzen oder kleinen Unternehmen eingesetzt wird.

Das Ethernet wurde durch die Arbeitsgruppe 802.3 des Institute of Electrical and Electronics Engineers (IEEE) in unterschiedlichen Versionen spezifiziert und normiert. Innerhalb des von der ISO standardisierten Open Systems Interconnection Reference Model (OSI), welches die Designgrundlage von Kommunikationsprotokollen in Rechnernetzen bildet, deckt Ethernet die untersten zwei Schichten ab. In der untersten Schicht, dem Physical Layer, definiert Ethernet die zu verwendenden Kabel, Anschlüsse und wie die Übertragung auf Bit-Ebene zu erfolgen hat. Für den Data Link Layer, der zweiten Schicht im OSI Schichtenmodell, wird durch Ethernet die Struktur des Datenpaketes definiert [16].

In der weit verbreiteten 802.3 Unternorm 100BASE-TX wird für die Signalübertragung mindestens ein **CAT-5** in UTP-Ausführung vorgesehen. Dieses besteht aus einem verdrehten Adernpaar pro Übertragungsrichtung. Für den Anschluss sind Modularstecker und -buchsen vom Typ **8P8C**, die im Allgemeinen als **RJ-45** bezeichnet werden, spezifiziert. Damit können Netzsegmente von einer maximalen Länge von 100 m bei einer Datenübertragungsrate von  $100\text{MBit/s}$  überbrückt werden.

Die Übertragung der Datensignale erfolgt, wie auch bei allen anderen zuvor betrachteten Schnittstellen, im Basisband, d.h. ohne Modulation. Da eine lange Aufeinanderfolge von binären **NULLEN** und **EINSEN** im Logikpegel die Taktrückgewinnung erschwert, wird

der Logikpegel zunächst einer 4B5B-Codierung unterzogen. Aus einem 4-Bit Datenblock wird so ein 5-Bit Codeblock gebildet. Mit diesem einen zusätzlichen Bit stehen nun auch Sondersymbole zur Verfügung, die unter anderem der Synchronisierung und Signalisierung der Übertragung dienen. Durch die 4B5B-Codierung erhöht sich das zu übertragende Datenvolumen allerdings auf 125 %. Dieser binäre Datenstrom wird anschließend mit dem ternären Codierungsverfahren Multilevel Transmission Encoding - 3 levels (MLT-3) codiert. Der so entstandene Signalstrom enthält nun drei Spannungspegel. Dadurch reduziert sich die zur Übertragung notwendige Bandbreite auf 25 % gegenüber einer rein binären Übertragung. Ausgehend von einer Datenübertragungsrate von  $125\text{ MBit/s}$  sinkt die erforderliche Bandbreite auf 31,25 MHz [13].

Um bei auftretender Sender-Konkurrenz die Kollision von Datenpaketen zu erkennen, kommt hier das Verfahren Carrier Sense Multiple Access / Collision Detection (CSMA/CD) zum Einsatz. Dabei lauscht jeder Teilnehmer auf dem Übertragungsmedium, ob es durch eine andere Übertragung belegt ist. Jeder Teilnehmer kann dann mit dem Senden seines Datenpaketes beginnen, sobald er das Ende einer anderen Übertragung erkannt hat. Kommt es nun zum gleichzeitigen Senden mehrerer Teilnehmer, so wird die Übertragung unverzüglich von allen abgebrochen. Teilnehmer mit einer unterbrochenen Sendung warten dann einen zufällig langen Zeitintervall, der sogenannten Backoff Time, und versuchen dann erneut, ihr Datenpaket zu versenden. Damit kann allerdings nicht garantiert werden, nach welcher Zeit ein Datenpaket sein Ziel erreicht. Das Ethernet ist damit nicht echtzeitfähig. Die maximal erreichbare Datenübertragungsrate wird durch diese Kollisionserkennung ebenfalls stark beeinflusst. Sind nur wenige Teilnehmer am Netz aktiv beteiligt, so treten auch Kollisionen mit geringer Wahrscheinlichkeit auf. Unter günstigen Umständen werden so ca. 70 % der maximal möglichen Übertragungsgeschwindigkeit des Mediums erreicht. Wird das Netz hingegen durch das Vorhandensein vieler aktiver Teilnehmer belastet, so treten auch vermehrt Kollisionen auf. Dies führt zu einer Verminderung der real nutzbaren Übertragungskapazität von bis zu 30 % der maximalen Übertragungsgeschwindigkeit.

Ein Datenpaket im Ethernet hat gemäß der Spezifikation für den Typ 100BASE-TX einen Mindestumfang von 64 Bytes. Diese Länge ergibt sich aus der Notwendigkeit, dass alle Teilnehmer innerhalb des Netzes Kollisionen erkennen müssen, um eine Sendewiederholung zu initiieren. Bedingt durch die Parameter Übertragungsrate, maximal zulässige Übertragungsbereichweite und der Ausbreitungsgeschwindigkeit der Signale im Medium wurde dieser Wert spezifiziert.

Dem Ethernet-Datenpaket geht dabei stets eine 8 Byte lange Präambel voraus. Sie besteht aus 7 Bytes der hexadezimalen Wertigkeit 55h und dem Start Frame Delimiter (SFD). Die Präambel diene ursprünglich den am Netzwerk beteiligten Geräten zur Synchronisierung und wird zur Definition der Mindestlänge des Datenpaketes nicht mitgezählt. Das eigentliche Datenpaket beginnt dann mit den beiden Media Access Control-Adressen (MAC), wobei die Erste das Ziel bestimmt und die Zweite für den Absender steht. Die MAC-Adresse besteht aus 6 Bytes und sollte innerhalb eines LAN's eindeu-

tig und einzigartig sein. Durch Verwendung der MAC-Adresse `FF:FF:FF:FF:FF:FF` als Ziel wird das Datenpaket an alle Teilnehmer des LAN versandt. Diese Adresse wird als Broadcast MAC Adresse bezeichnet. Nach der Absender-MAC-Adresse folgt optional das 4 Bytes umfassende Virtual Local Area Network Feld (VLAN). Die Kennzeichnung des VLAN erfolgt durch den Wert der ersten beiden Bytes mit `8100h`. Durch VLAN lassen sich auf der dritten Schicht des OSI-Modelles, dem Network Layer, logische Netze innerhalb eines LAN bilden. Das zwei Byte umfassende Typ-Feld definiert das verwendete Protokoll auf höheren Schichten für die nun folgenden Nutzdaten. Die zu übermittelnden Daten können einen Umfang von 0-1500 Bytes haben und werden durch den Parameter Maximum Transmission Unit (MTU) festgelegt. Die MTU beschreibt die maximal zulässige Länge eines von der darüber liegenden OSI Schicht zu versendenden Datenpaketes über Ethernet als Nutzlast. Zwar werden von den meisten Netzwerkgeräten auch Nutzlastgrößen von deutlich mehr als 1500 Bytes unterstützt, diese Pakete bilden die sogenannten Jumbo Frames, entsprechen jedoch nicht der Norm nach 802.3. Unterschreitet die zu übertragende Datennutzlast die Länge von 46 Bytes (ohne Verwendung des VLAN-Feldes) bzw. 42 Bytes (bei Verwendung des VLAN Feldes), so werden Padding Bytes im Pad-Feld hinzugefügt, um die Mindestpaketlänge von 64 Bytes zu erreichen. Über das gesamte Ethernet-Datenpaket, beginnend mit der Ziel-MAC-Adresse bis hin zum PAD-Feld, wird nun eine 32-Bit CRC Prüfsumme berechnet und sie bildet so den letzten 4 Byte umfassenden Teil des Datenpaketes. Der schematische Aufbau des Ethernet-Datenpaketes kann der Abbildung 2.11 entnommen werden.

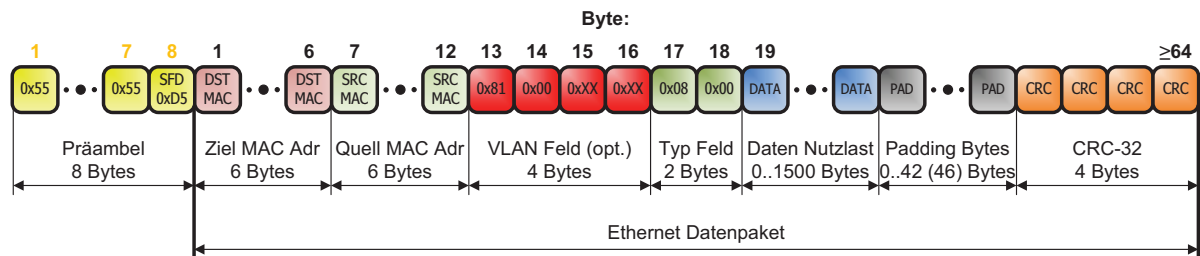


Abbildung 2.11: Aufbau des Ethernet-Datenpaketes

Trotz fehlender Echtzeitfähigkeit haben sich die auf das Ethernet basierenden Netzwerkstrukturen in weite Bereiche der Industrie, der Telekommunikation und auch im privaten Umfeld durchgesetzt. Auf Basis des Ethernet entstehen komplexe und leistungsfähige Netzstrukturen, die dem Datenaustausch verschiedenster Teilnehmer durch Ausnutzung der im OSI-Modell übergeordneten Schichten dienen. Durch den Einsatz neuer Übertragungsmedien, wie zum Beispiel Glasfaserkabel oder CAT6/7-Kabel, ist ein Ende der Steigerung der Übertragungsgeschwindigkeit noch nicht abzusehen. Aktuelle Unternormen der IEEE Arbeitsgruppe 802.3 erreichen nominale Datenübertragungsraten von  $10\text{GBit/s}$ .

## 3 Konzept der Multifunktions-Adapterplatine

### 3.1 Hardwarekonzept

#### 3.1.1 Grundkonzept

Die Hauptaufgabe der Multifunktions-Adapterplatine ist es, Daten aus verschiedenen Quellen entgegenzunehmen und über einen zentralen Kanal an das Ziel weiterzureichen. Um die Aufgabe des multifunktionalen Datenkonverters zu erfüllen, sind unterschiedliche Schnittstellentypen physisch und logisch zu realisieren. Folgende Schnittstellen für den eingehenden Datenstrom gilt es hierbei zu unterstützen und zu realisieren:

- serielle Schnittstellen auf UART-Basis (RS-232/422/485)
- SPI
- I<sup>2</sup>C
- CAN
- USB

Die Wahl der Schnittstelle für den abgehenden Datenstrom fällt dabei auf Ethernet. Mit der Unternorm 10BASE-TX steht ein hinreichend schnelles und kostengünstiges Übertragungsmedium zur Verfügung, die eingehenden Daten in gebündelter Form durch Verwendung von Protokollen höherer OSI Schichten zu versenden. Da der geplante Anwendungsbereich es vorsieht, die Multifunktions-Adapterplatine in kleine Netze mit geringer Auslastung zu integrieren, ist das Fehlen der Echtzeitfähigkeit von Übertragungen durch das Ethernet vernachlässigbar.

Das Empfangen der Daten über die verschiedenen Schnittstellen, die Zwischenspeicherung und das Senden dieser Daten erfolgt durch eine MCU. Das zur Verfügung stehende *mbed*-Mikrocontrollermodul stellt ausreichend Rechenleistung und Arbeitsspeicher bereit, um diese Aufgabe zu erfüllen. Ebenso zeigt sich dieses Modul sehr kontaktfreudig, da alle oben genannten Schnittstellen über die Input-/Outputpins bereits auf den Data Link Layer des OSI Schichtenmodells zur Verfügung stehen.

Die Ein- und Ausgänge des *mbed*-Moduls sind dabei nicht fest definiert. Viele davon sind in der Lage, mehrere Funktionalitäten wahrzunehmen und damit unterschiedliche Schnittstellen zu unterstützen. Dadurch ist eine Modularisierung einiger Schnittstellen möglich, die als Module, den Breakout-Boards, realisiert werden können. Dies macht die

Einführung von Steckplätzen, den sogenannten Breakout-Sockets, auf der Hauptplatine notwendig und sinnvoll. Als tragendes Element wird die Hauptplatine im Folgenden als Base-Board bezeichnet. Je nach Anwendung kann das Base-Board so mit den benötigten Schnittstellen ausgestattet werden. Diese übernehmen dann die Aufgabe zum Senden und Empfangen des Leitungsspannungspegels und realisieren so die erste Schicht des OSI Schichtenmodells. Diese Modularität lässt der Multifunktions-Adapterplatine ausreichend Spielraum für zukünftige Entwicklungen und sich ändernde Einsatzzwecke.

Die Hauptspannungsversorgung der Multifunktions-Adapterplatine sieht dabei zwei unterschiedliche und beiläufige Vorgehensweisen vor. Die Erste basiert auf die Versorgung des Base-Board und der Breakout-Boards mit Hilfe eines geregelten DC/DC-Wandlers. Dieser wandelt die Eingangsgleichspannung in die primäre Versorgungsspannung von  $+5V$  mit der alle Komponenten betrieben werden. Als Alternative ist die Verwendung eines PoE-Moduls zur Generierung der Primärspannungsversorgung vorgesehen. Durch Verwendung geeigneter Switches/Router im LAN erfolgt so die Spannungsversorgung über die Leitung der Ethernet Schnittstelle. Damit ist direkt keine weitere Einspeisung einer Versorgungsspannung zur Multifunktions-Adapterplatine notwendig.

Um die Leistungsaufnahme der Multifunktions-Adapterplatine so gering wie möglich zu halten, wird die Auswahl der entsprechenden elektronischen Bauelemente so getroffen, dass eine Versorgungsspannung von  $3,3V$  ausreichend ist. Dies hat zum Vorteil, dass keine Pegelanpassung der digitalen Signale von und zum *mbed*-Modul zu erfolgen hat. Somit wird auf dem Base-Board neben der Hauptspannungsversorgung auch eine Versorgungsspannung von  $3,3V$  eingesetzt.

Bedingt durch die Vorgabe zur Verwendung stromsparender elektronischer Komponenten, sind Base-Board und Breakout-Boards größtenteils in Bauweise der Surface-Mounted Devices (SMD) konzeptioniert und realisiert. Dies ermöglicht die Platzierung vieler elektronischer Komponenten bei minimalen Bedarf an Platinenfläche.

#### 3.1.2 Das *mbed*-Mikrocontrollermodul

Die Basis der Multifunktions-Adapterplatine bildet das *mbed*-Mikrocontrollermodul. Es stellt die Kommunikation und Funktion der physischen Schnittstellen sicher. Auf dem *mbed* kommt die MCU LPC1768 von der Firma NXP zum Einsatz. Diese MCU basiert auf der ARM Cortex-M3 Plattform und ist in Harvard-Architektur gehalten. Mit einer Taktfrequenz von  $96MHz$  und dem integrierten Static Random Access Memory (SRAM) von  $64KiB$  bietet diese 32-Bit MCU genug Potential zur Bewältigung komplexer Datenverarbeitungsaufgaben.

Die MCU selbst enthält bereits eine Vielzahl von Peripheriekomponenten, die für die Realisierung serieller Schnittstellen zur Verfügung stehen. Für die Abbildung der Daten der Ethernetschnittstelle auf den Physical Layer enthält das *mbed*-Modul bereits die ent-

sprechende physikalische Schnittstelle, realisiert als elektronisches Bauelement auch als PHY bezeichnet.

Das *mbed*-Modul ist als vollständiges und kompaktes Bauteil in einem 40-poligen Dual In-line Package (DIP) ausgeführt. Durch die mittels zweier Stiftleisten im Rastermaß von  $2,54\text{mm}$  nach außen geführten Kontakte, lässt es sich ohne Verlöten auf einen Sockel platzieren. All jene Ein-/Ausgabekontakte, die nicht bereits auf dem *mbed*-Modul mit Peripheriebausteinen verbunden sind, können dabei mehrere Funktionen wahrnehmen. Dies betrifft die Pins 5 – 30, die neben der Realisierung serieller Schnittstellen, auch in jedem Fall als digitale Ein- und Ausgänge verwendet werden können. Über die Pins 15-20 kann der in der MCU integrierte 6-Kanal ADC verwendet werden. Er bietet eine Auflösung von 12 Bit. Abbildung 3.1 zeigt die Kontaktbelegung des *mbed*-Moduls und die typische Funktion des jeweiligen General Purpose Input/Output (GPIO) Pins. Alle GPIOs sind für die Verwendung eines CMOS-Logikpegels von  $+3,3\text{V}$  ausgelegt.

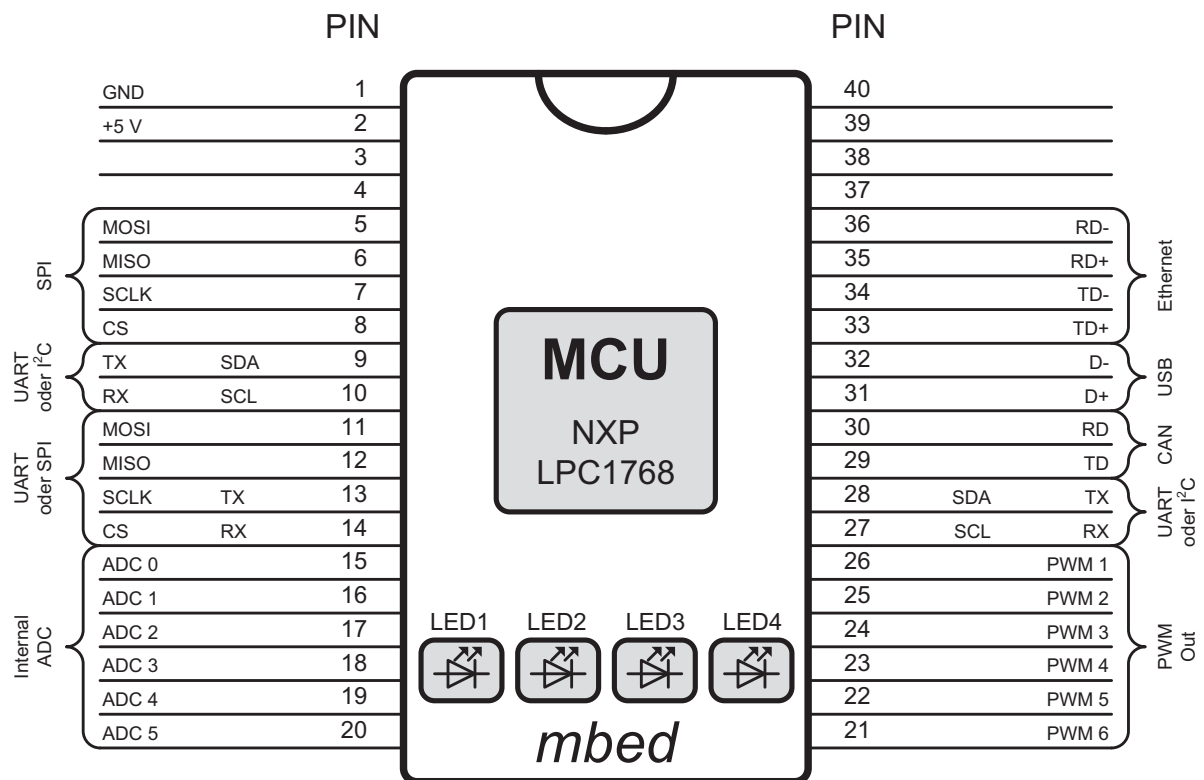


Abbildung 3.1: Kontaktbelegung des *mbed*-Mikrocontrollermoduls

Die Programmierung des *mbed*-Mikrocontrollermodul erfolgt über die Mini-USB Schnittstelle. Bei der Verbindung des *mbed*-Moduls mit den PC verhält es sich wie ein USB Massenspeicher. Das nach dem Compilieren erhaltene Binärfile mit der Extension `bin` wird dann einfach in den Flashspeicher kopiert. Bei Neustart des *mbed*-Moduls wird das zuletzt gespeicherte Binärfile aufgerufen und ausgeführt.

Die Spannungsversorgung des *mbed* erfolgt über Pin 2. Die über das Base-Board eingespeiste Versorgungsspannung von  $+5\text{V}$  liegt im zulässigen Bereich von  $+4,5\text{V}$  bis  $+9\text{V}$ .

Ein weiteres Ausstattungsmerkmal des *mbed*-Mikrocontrollermoduls ist das Vorhandensein von vier LEDs, die für Zwecke der Signalisierung genutzt werden können. Ein Taster auf der Oberseite des Moduls löst das RESET für die MCU aus.

### 3.1.3 Das Base-Board

Das Base-Board stellt das physische Trägermedium dar, dass die Gesamtfunktionalität der Multifunktions-Adapterplatine ermöglicht. Diese Basisplatine ist Träger aller modularen Komponenten, wie dem *mbed*-Mikrocontrollermodul und den Breakout-Boards. Auf dem Base-Board befindet sich die Spannungsversorgung und all jene Schnittstellen, die durch das *mbed* eindeutig vorgegeben sind. Abbildung 3.2 zeigt hier schematisch den konzeptionellen Aufbau des Base-Boards.

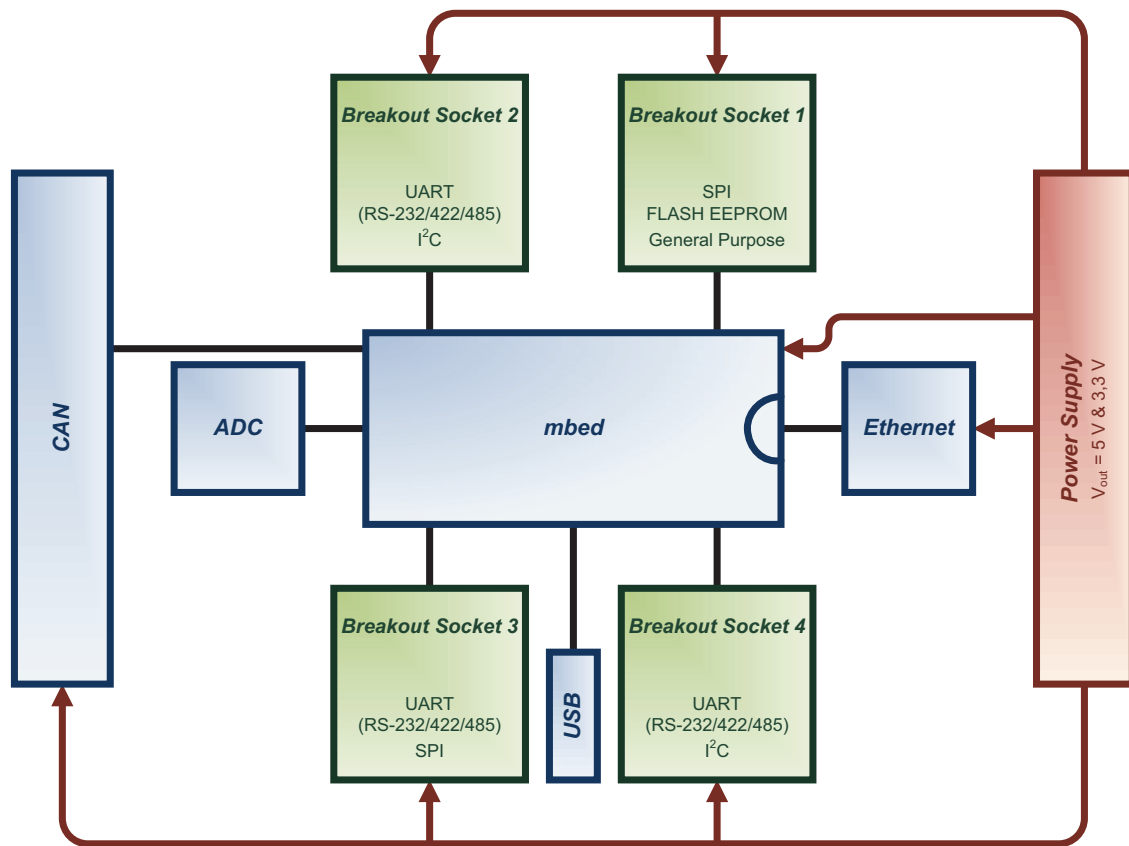


Abbildung 3.2: Schematischer Aufbau des Base-Boards

Um alle Komponenten der Multifunktions-Adapterplatine aufnehmen zu können, weist das Base-Board die Abmessungen einer Euro-Platine in der Dimension von  $160\text{mm} \times 100\text{mm}$  auf. Durch zwei parallel montierte 20-polige Buchsenleisten erfolgt die Aufnahme des *mbed*-Moduls im Zentrum des Base-Boards.

In direkter Nachbarschaft zum *mbed* befinden sich jeweils zwei 9-polige Sockel zur Aufnahme der Breakout-Boards. Durch die Nähe zum Mikrocontrollermodul wird die Länge

der Datenleitungen möglichst kurz gehalten und das Einbringen von Störungen vermieden. Das Base-Board ist so mit insgesamt vier genormten Breakout-Boards bestückbar.

Die vom *mbed*-Modul fest vorgegebenen Schnittstellen sind als feste Elemente auf dem Base-Board realisiert. Folgende Schnittstellen sind auf dem Base-Board enthalten:

- Ethernet
- CAN
- USB

Für die Realisierung der Ethernet Schnittstelle kommt hier eine spezielle RJ-45 Buchse zum Einsatz. Durch die in diesem Anschluss integrierten magnetischen Übertrager wird eine Isolation zwischen Base-Board und Ethernetleitung erreicht. Eine solche Buchse wird als Mag45 bezeichnet. Durch die Maßgabe, die Hauptspannungsversorgung über das Verfahren PoE zu unterstützen, kommt hier eine spezielle PoE-taugliche Mag45 Variante zum Einsatz. Da der Ethernet-PHY bereits Bestandteil des *mbed*-Moduls ist, werden die vier Ethernet Datenleitungen direkt mit datenführenden Kontakten der Mag45 verbunden.

Die CAN-Schnittstelle wird ebenso durch einen magnetischen Übertrager vom Base-Board isoliert. Dieser wird ebenso wie der CAN-Treiber-IC mit einer Versorgungsspannung von 3,3V versorgt, um die Leistungsaufnahme zu minimieren. Um die Multifunktions-Adapterplatine innerhalb eines CAN-Bussystems möglichst flexibel zu platzieren, ist der parallele Terminierungswiderstand durch einen Jumper zu- bzw. abschaltbar. Die Kontaktierung des mechanischen Anschlusses erfolgt nach den Richtlinien des CANopen Standards. Die Anschlüsse sind so gewählt und beschaltet, dass die Integration der Multifunktions-Adapterplatine sowohl durch Stichleitungen als auch durch Bildung einer Daisy-Chain Bustopologie, wie in Abbildung 3.3 gezeigt, erfolgen kann. Die CAN-Schnittstelle ist entsprechend der Spezifikation für Übertragungsgeschwindigkeiten von bis zu 1MBit/s ausgelegt.

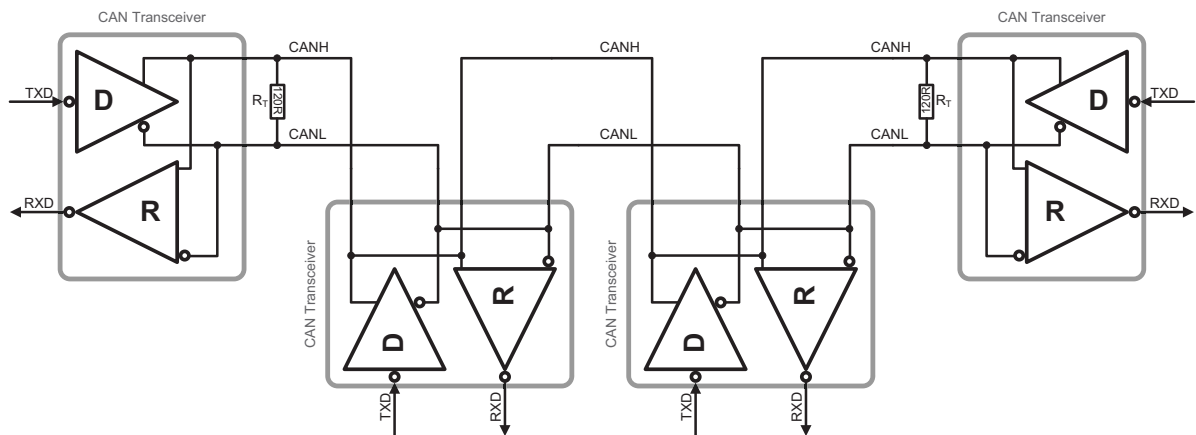


Abbildung 3.3: CAN-Bus in Daisy-Chain Bustopologie

Die USB-Schnittstelle ist aufgrund der Beschaltung durch das *mbed*-Modul ausschließlich als USB Device nutzbar, wodurch sich das Anwendungsfeld dieser Schnittstelle

einengt. Für zukünftige Anwendungen wird der mechanische Anschluss daher als 4-polige Stiftleiste konzipiert.

Auf dem Base-Board sind weitere Stiftleisten als analoge Eingänge für den internen ADC der MCU vorgesehen. Die Kanäle 1 bis 4 des ADC stehen zur Wandlung von analogen Signalen in zeitdiskrete Werte zur Verfügung.

Da viele SPI Komponenten eine **RESET**-Leitung benötigen, ist die Generierung dieses Signals für die SPI fähigen Breakout-Sockets 1 und 3 vorgesehen. Die Ansteuerung für die **RESET**-Signalерzeugung geschieht über zwei GPIOs des *mbed*-Moduls aus dem Kontingent der nicht benötigten Anschlüsse.

Für die Hauptspannungsversorgung der Multifunktions-Adapterplatine sind zwei unterschiedliche Konzepte vorgesehen. Das Base-Board kann mit der Einen oder der Anderen oder auch mit beiden Varianten bestückt werden. Generell ist daher eine Umschaltmöglichkeit zur Wahl der Primärspannungsversorgung vorhanden. Die erste Variante sieht durch Nutzung eines Gleichspannungswandlers die Versorgung des Base-Boards durch eine Eingangsgleichspannung im Bereich von  $9 - 36V$  vor. Alternativ dazu kann durch das PoE-Verfahren über die Ethernet Schnittstelle die Versorgungsspannung zugeführt werden. Durch das auf dem Base-Board enthaltene PoE-Modul kann die Eingangsspannung dann in die notwendige Ausgangsspannung transferiert werden. Beiden Konzepten gemein ist, dass sie eine Ausgangsspannung von  $V_{OUT} = +5V$  zur Versorgung der Multifunktions-Adapterplatine bei einer maximalen Last von ca.  $10W/2A$  liefert. Um die Komponenten der Primärspannungsversorgung vor Überlastung zu schützen, kommt hier zentral eine austauschbare SMD-Sicherung mit der maximalen Strombelastung von  $2A$  zum Einsatz.

Die Versorgung der Komponenten der Multifunktions-Adapterplatine sieht eine duale Spannungsversorgung von  $+5V$  und  $+3,3V$  vor. Dies bedingt den Einsatz eines linearen Spannungsreglers, der durch möglichst geringen Schaltungsaufwand die zur Verfügung stehende Versorgungsspannung von  $+5V$  in  $+3,3V$  umsetzt. Für beide Spannungspotentiale werden entsprechende mechanische Abgreifmöglichkeiten vorgesehen, um eventuelle Erweiterungen zu unterstützen. Die auf den Versorgungsleitungen anliegende Spannung wird durch den Einsatz von LEDs signalisiert. Als Festlegung gilt hier, dass eine rote LED eine Spannung von  $+5V$  signalisiert. Das Leuchten einer gelben LED steht für das Vorhandensein der Versorgungsspannung von  $+3,3V$ .

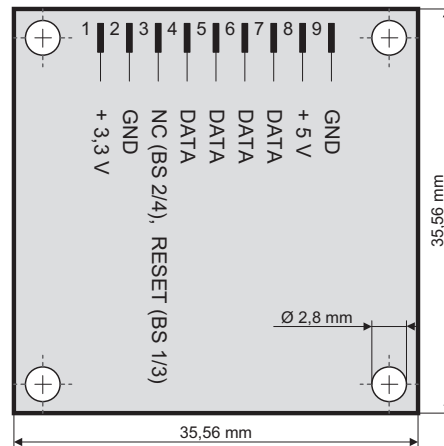
#### 3.1.4 Modularisierung durch Breakout-Boards

Die Breakout-Boards stellen die Realisierung der jeweiligen Schnittstelle auf der untersten Schicht des OSI Schichtenmodells dar. Für einen speziellen Zweck konzipiert, finden sie auf dem entsprechendem Breakout-Socket des Base-Boards Platz. Durch diese Module ist es möglich, die Multifunktions-Adapterplatine speziell für den geforderten Aufgabenbereich mit den benötigten Schnittstellen zu bestücken und zu betreiben. Im Rahmen dieser

Masterarbeit werden Breakout-Boards für die Schnittstellen RS-232, RS-422 und RS-485 konstruiert. Drei Prototypen für die RS-232 Schnittstelle werden dabei realisiert.

Für die Konzeption der Abmessungen der Breakout-Boards besteht die Maßgabe: „So groß wie nötig, so klein wie möglich !“. Schließlich gilt es, vier dieser Module auf den Breakout-Sockets des Base-Boards gleichzeitig unterzubringen. Die Abmessungen eines Breakout-Boards werden mit einer quadratischen Form der Seitenlänge von  $35,56\text{mm}$  festgelegt. Mit dieser Dimensionierung wird eine Platzierung in direkter Nähe des *mbed*-Moduls möglich. Im Rahmen der Konstruktion der oben genannten Schnittstellen hat sich gezeigt, dass die zur Verfügung stehende Fläche ausreichend ist.

Die unterschiedlichen Typen von Breakout-Boards besitzen eine einheitliche Belegung des als 9-polige Stiftleiste ausgeführten Anschlusses. Diese Stiftleiste stellt die mechanische und elektrische Verbindung zu den Breakout-Sockets des Base-Boards her. Die beiden zur Verfügung stehenden Versorgungsspannungen von  $+3,3\text{V}$  und  $+5\text{V}$  sind stets auf den Pins 1 und 8 verfügbar. Das Massepotential steht an den Pins 2 und 9 bereit. Die Pins 4 bis 7 sind über Leiterbahnen auf dem Base-Board mit dem *mbed*-Modul verbunden und realisieren die entsprechende Funktionalität der Schnittstelle. Die RESET-Leitung der SPI fähigen Breakout-Sockets 1 und 3 befindet sich jeweils auf Pin 3. Abbildung 3.4 zeigt hier die generische Pinbelegung eines Breakout-Boards.



**Abbildung 3.4:** Generischer Aufbau eines Breakout-Boards

Der Aufbau der Breakout-Boards für die RS-232/422/485 Schnittstellen erfolgt dabei in gleicher Weise. Der obere Teil der Platine enthält neben der Kontaktierung zum Base-Board die Ansteuerung des DC/DC-Konverters. Denn auch hier werden die Systemkomponenten des Base-Boards von der busseitigen Schnittstellenkomponente isoliert. Das kontrollierte Ein- bzw. Ausschalten der Spannungsversorgung über einen GPIO der MCU minimiert die Leistungsaufnahme durch Deaktivierung nicht benötigter Baugruppen. Durch eine zeitversetzte Inbetriebnahme eines jeden Breakout-Boards verringern sich ebenso die Einschaltströme.

Da für die Schnittstellen RS-232, RS-422/485 keine speziellen Anschlüsse spezifiziert sind, werden generische Anschlüsse verwendet. Bei Einbau der Multifunktions-Adapterplatine in ein Gehäuse können so die benötigten Signalleitungen mit den am Gehäuse angebrachten Buchsen/Stecker verbunden werden.

Der Anschluss des RS-232 Breakout-Boards erfolgt über einen 10-poligen Wannenstecker. Durch Einsatz eines konfektionierten Flachbandkabels mit 10-poliger Wannenbuchse werden die kontaktierten Leitungen nach Außen geführt. Hier werden die Leitungen beispielsweise mit einer 9-poligen D-SUB Buchse/Stecker verbunden. Um die Verwendung von NULL-Modem als auch 1:1 kontaktierter Leitungskabel zu ermöglichen, kann die Belegung der Anschlüsse für die Signale RX und TX auf dem RS-232 Breakout-Board durch zwei Jumper getauscht werden.

Für die Kontaktierung der RS-422 und RS-485 Breakout-Boards mit den Leitungskabeln sind Printklemmen vorgesehen. Die 6-polige Klemme des RS-422 Breakout-Boards führt die Signale A, B, Y und Z nach außen. Die beiden Signale A und B des RS-485 Breakout-Boards werden durch eine 4-polige Printklemme mit den Leitungskabeln verbunden. Der parallele Terminierungswiderstand lässt sich bei beiden Schnittstellen durch einen Jumper aktivieren bzw. deaktivieren.

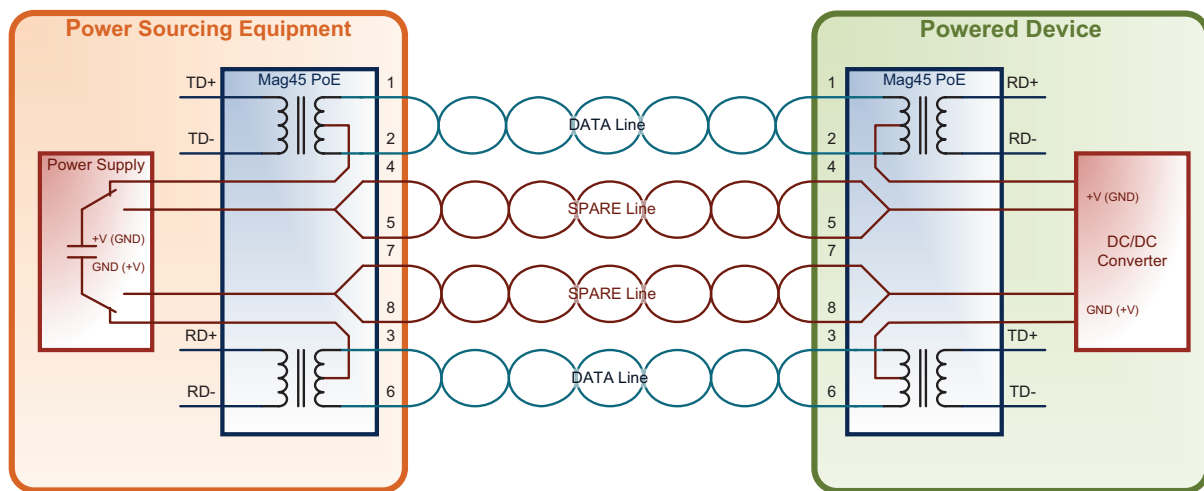
Für eine mechanisch stabile Platzierung der Breakout-Boards auf dem Base-Board werden jeweils vier Abstandhalter verwendet. Sie halten das Breakout-Board in einer waagerechten Position zum Base-Board. Die auf gleicher Position zu den Abstandhaltern liegenden Bohrungen im Base-Board ermöglichen durch Verschrauben eine mechanisch feste Verbindung zwischen Breakout-Board und Base-Board.

#### 3.1.5 Spannungsversorgung durch Power over Ethernet

Eine Möglichkeit, die Multifunktions-Adapterplatine mit Spannung zu versorgen, geschieht direkt über das Ethernetkabel. Dieses Verfahren wurde 2003 durch den Standard IEEE 802.3af spezifiziert. Ziel der Einführung dieser Norm ist es, über das Ethernetkabel neben dem Datenverkehr auch die Spannungsversorgung von Endgeräten zu realisieren. Damit können Voice over IP (VoIP) Telefone, Wireless Local Area Network (WLAN) Zugangsknoten und andere Geräte über ein und das selbe Kabel sowohl mit Daten als auch mit Energie versorgt werden.

Für das Einbringen der Versorgungsspannung sind keine speziellen Kabel nötig. Die in der jeweiligen Infrastruktur vorhandenen Kabelleitungen vom Typ CAT-5/6 können weiterhin verwendet werden. Das PoE Verfahren sieht zwei Möglichkeiten zur Übertragung der Versorgungsspannung vor. Die erste Möglichkeit nutzt die Tatsache aus, dass im Ethernetkabel der Kategorie 5 und 6 unter Nutzung der 100BASE-TX Übertragung von den vier vorhandenen Leitungsadern nur zwei zur Datenübertragung genutzt werden. Die ungenutzten Adern der Pins 4/5 und 7/8 können mit jeweils einer Polarität der zu übertragenden Spannung gespeist werden. Die eingespeiste Gleichspannung liegt in einem

Bereich von  $+44V$  bis  $+57V$ . Alternativ dazu können aber auch die zur Datenübertragung vorgesehen Leitungsadern der Pins 1/2 und 3/6 zur Einspeisung der Gleichspannung genutzt werden. Durch den Einsatz von Transformatoren an beiden Enden einer Datenverbindung erfolgt das Einbringen und Abgreifen des Gleichspannungsanteils. Das die Versorgungsspannung einspeisende Gerät wird als PSE bezeichnet und hauptsächlich als PoE Switch realisiert. Das PD stellt den Verbraucher dar, der durch die eingebrachte Gleichspannung versorgt wird. Gemäß der Spezifikation muss ein PD in der Lage sein, beide Arten der Gleichspannungsübertragung zu unterstützen. Abbildung 3.5 zeigt die Spannungsversorgung des PD durch ein PSE [18].



**Abbildung 3.5:** Spannungsversorgung eines PD durch ein PSE

Die erste Hürde für das Verwerten der übertragenen Versorgungsspannung durch das PD ist es, die Polarität der Spannung zu ermitteln. Die Spezifikation des PoE-Verfahrens enthält keine Vorgabe, auf welchen Adern welche Polarität die Gleichspannung übertragen wird. Als sehr praktikable und preiswerte Lösung hat sich hier der Einsatz eines Brückengleichrichters bewährt.

Um sicherzustellen, dass ein nicht-PoE fähiges Endgerät nicht durch die eingespeiste Spannung beschädigt oder zerstört wird, beginnt jetzt die Phase der Signaturerkennung. In dieser Phase wird durch das PSE eine Spannung von  $2,7V$  bis  $10,1V$  in das Ethernetkabel eingespeist. Durch Strommessung der PSE wird das Vorhandensein des parallel geschalteten Signaturwiderstandes von  $25k\Omega$  im PD detektiert. Für ein PD gilt außerdem die Bedingung, dass die Eingangskapazität kleiner  $150nF$  betragen muss. Auch dies wird in dieser Phase durch das PSE überprüft.

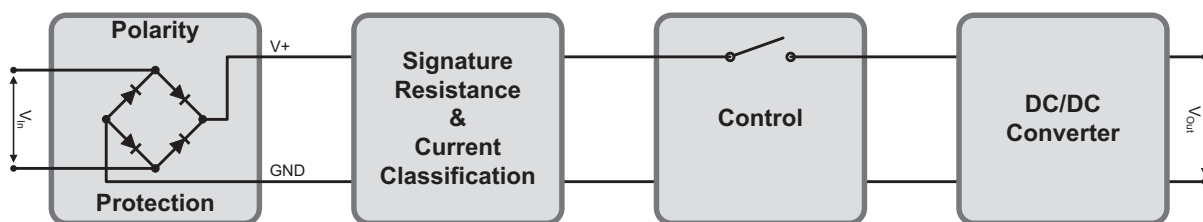
Verläuft die Signaturerkennung positiv, es wurde ein PoE fähiges Netzwerkgerät detektiert, folgt nun die Klassifizierungsphase. In dieser Phase wird durch das PSE die maximal zu erwartende Leistungsaufnahme des PD bestimmt. Dies geschieht wiederum mit einer Strommessung nachdem eine Gleichspannung im Bereich von  $14,5V$  bis  $20,5V$  durch das PSE eingespeist wurde. Das PD wird dann einer der in Tabelle 3.1 aufgeführten

Leistungsklassen zugeordnet und das PSE limitiert dementsprechend den maximalen Leistungsstrom. Diese Klassifizierung erweist sich als nützlich beim Energiemanagement in größer dimensionierten Switches [17].

| Klasse | gemessener Strom / mA | max. PD Leistung / W | Verwendung |
|--------|-----------------------|----------------------|------------|
| 0      | 0..4                  | 0,44..12,95          | Standard   |
| 1      | 9..12                 | 0,44..3,84           | optional   |
| 2      | 17..20                | 3,84..6,49           | optional   |
| 3      | 26..30                | 6,49..12,95          | optional   |
| 4      | 36..40                | Reserviert           | zukünftig  |

**Tabelle 3.1:** PD Leistungsklassen

Sind alle Phasen der PD-Überprüfung positiv verlaufen wird die Spannungseinspeisung seitens des PSE in das Ethernetkabel gestartet. Durch einen nachgeschalteten DC/DC-Wandler, der sich in der Baugruppe der Spannungsversorgung des PD befindet, kann dann die eigentliche Versorgungsspannung des PD erzeugt werden. Eine schematische Darstellung der Spannungsversorgung eines PD kann der Abbildung 3.6 entnommen werden.



**Abbildung 3.6:** Blockschaltbild eines PoE-Moduls

Das in der Multifunktions-Adapterplatine zum Einsatz kommende PoE-Modul arbeitet genau nach dem oben beschriebenen Prinzip. Das hier verwendete Modul Ag9605-2BR der Firma Silvertel ist ein Klasse 0 Gerät und versorgt sowohl Base-Board als auch die Breakout-Boards mit einer Versorgungsspannung von +5V bei einer maximalen Leistungsaufnahme von ca. 10W.

Mit der Spezifikation des PoE-Verfahrens steht ein leistungsfähiges Mittel zur Verfügung, die zur Datenübertragung über Ethernet mit dem Internet verbundenen Endgeräte über die selbe Leitung in gleicher Weise auch mit entsprechender Energie zu versorgen. Durch den Austausch der Switches/Hubs gegen die als PSE operierenden Netzverteiler können bestehende Netzwerke ohne kostspielige infrastrukturelle Änderung auf diese Möglichkeit der genormten Spannungsversorgung erweitert werden.

### 3.1.6 Galvanische Trennung

Die Multifunktions-Adapterplatine ist typischerweise über die Schnittstellen mit einer Vielzahl von anderen elektrischen Geräten verbunden. Die Leitungslängen können da-

bei von ein paar wenigen Metern bis zu tausend Meter betragen. Um die elektronischen Komponenten der Multifunktions-Adapterplatine vor den in die Leitungen eingebrachten Störungen zu schützen, ist eine galvanische Trennung zwischen jeder Schnittstelle und dem Base-Board unverzichtbar. Das Entkoppeln der elektrischen Potentiale schützt das Base-Board vor auf der Leitung entstehenden Überspannungsspitzen und verhindert die Ausbildung von Masseschleifen. Die galvanische Trennung ist eine Maßnahme, die EMV zu erhöhen.

Durch die Entkopplung verfügt jede einzelne Schnittstellenkomponente eines Bussystems über ein eigenes isoliertes Massepotential, wodurch die Ausbildung von Masseschleifen vermieden wird.

Weit verbreitete Verfahren zur galvanischen Trennung sind das Übertragen von Informationen mit Licht durch den Einsatz von Optokopplern und die Übertragung über ein Magnetfeld durch Transformatoren. Bei beiden Verfahren wird zwischen den elektronischen Komponenten ein sehr hoher Isolationswiderstand mit entkoppelnder Wirkung eingebracht. Zwischen Systemkomponente und isolierter Schnittstellenkomponente können sich die Massenpotentiale deutlich unterscheiden und werden durch die technische Spezifikation der entkoppelnden Komponente, der Isolationsspannung, begrenzt [15].

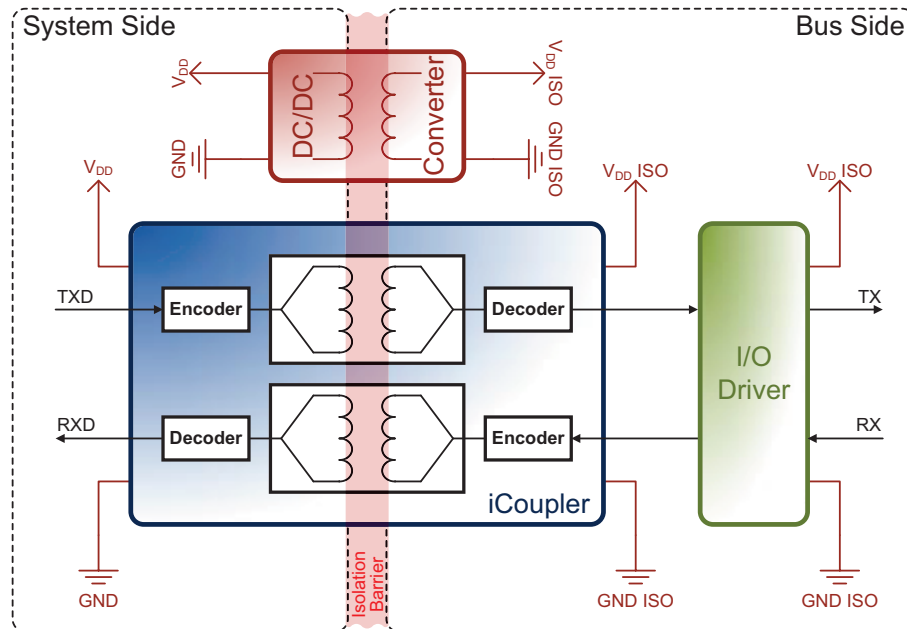
Lange Zeit galt der Einsatz von magnetischen Übertragern zur Isolation der Datenpfade als zu teuer und zu umständlich, da die Bauelemente relativ groß waren. Dies änderte sich mit dem Erscheinen von Mikrotransformatoren, die Anfang des letzten Jahrzehnts durch die Firma Analog Devices entwickelt wurden. Bei der als *iCoupler* bezeichneten Technologie werden die beiden Übertragungsspulen direkt übereinander in einem IC integriert. Dazwischen befindet sich eine hoch isolierende Schicht aus Polyimide. Der Datenstrom wird dabei nicht direkt über die Spulen übertragen. Ein Encoder am Dateneingang detektiert jeden Flankenwechsel und überträgt diese Information über den Mikrotransformator. Ein Decoder am Datenausgang rekonstruiert dann wieder das eigentliche Datensignal [8].

Bei der Realisierung der CAN- und der RS-232/422/485-Schnittstelle wird ein Magnetübertrager aus dieser *iCoupler* Serie als digitaler Signalisolator verwendet. Im Gegensatz zu den weit verbreiteten Optokopplern, zeichnen sich die *iCoupler*-Komponenten durch eine deutlich geringere Leistungsaufnahme bei sehr hoher Datenübertragungsrate aus. Bedingt durch die induktive Kopplung, stehen auch bidirektionale Isolatoren mit mehreren Ein- und Ausgängen zur Verfügung. Die Isolationsspannung beträgt hierbei 2500V.

Für die galvanische Trennung der Versorgungsspannung werden DC/DC-Wandler verwendet. Nach der Wandlung steht eine von der Systemkomponente unabhängige und isolierte Versorgungsspannung mit eigenem Potential zur Versorgung des Schnittstellentreibers und der isolierten Seite des *iCoupler* zur Verfügung. Die verwendeten DC/DC Wandler weisen eine Isolationsspannung von 1000V auf.

Die galvanische Trennung der Systemkomponente von der Schnittstellenkomponente spiegelt sich auch im Layout der einzelnen Platinen der Multifunktions-Adapterplatine

wieder. Die Isolation geschieht hier nicht ausschließlich durch die Verwendung entsprechender Bauelemente. Bereits in der Konstruktionsphase werden daher Isolationsbarrieren auf den einzelnen Printed Circuit Boards (PCB) vorgesehen, die eine entsprechende Isolierung sicherstellen. Eine schematische Darstellung der galvanischen Trennung kann der Abbildung 3.7 entnommen werden.



**Abbildung 3.7:** Grundschemata der Isolation der Systemkomponente von der Schnittstellenkomponente

Sowohl induktive Übertrager als auch Optokoppler besitzen dabei eine gewisse Kopplkapazität, die sich durch die Isolationsschicht bildet. Durch die daraus resultierende Hochpasscharakteristik können besonders hochfrequente Spannungsspitzen fast ungehindert die Isolationsbarriere passieren und elektrische Bauelemente der Systemkomponente beschädigen oder zerstören. Daher sind weitere Maßnahmen nötig, um dies zu verhindern [15].

### 3.1.7 Überspannungsschutz

Durch Blitze, elektrostatische Entladungen, Lastwechsel und induktive Schaltvorgänge können sich auf den Datenleitungen Impulse mit extrem hoher Amplitude bilden und ausbreiten. Die elektronischen Komponenten der Multifunktions-Adapterplatine können durch diese Transienten beschädigt oder zerstört werden und müssen daher in geeigneter Weise geschützt werden.

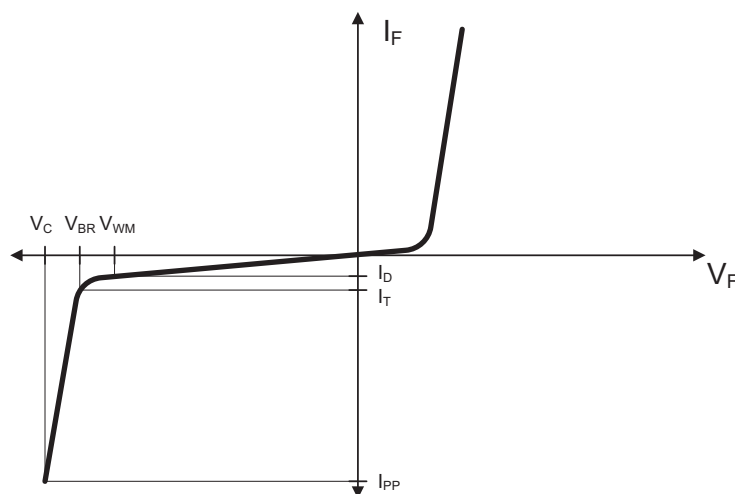
Die Spezifizierung des Standards IEC61000 legt Maßnahmen zur Sicherstellung der EMV fest. Drei Unterkapitel beschreiben Testprozeduren, die für die Klassifizierung der

Verträglichkeit gegenüber Transienten von Schnittstellenkomponenten von großer Bedeutung sind [5]:

- IEC 61000-4-2, Elektrostatische Entladungen (*engl.*: electrostatic discharge (ESD))
- IEC 61000-4-4, schnell aufeinanderfolgende Transienten (*engl.*: electrical fast transient/burst (EFT))
- IEC 61000-4-5, Zerstörfestigkeit (*engl.*: surge immunity)

Eine besondere Aufmerksamkeit gilt dabei den Transienten, die durch ESD hervorgerufen werden. Durch eine geeignete Auswahl an Bauelementen und ICs lässt sich hier bereits ein höchstmöglicher Schutz einzelner Komponenten erzielen. Auf der Multifunktions-Adapterplatine kommen daher nur Schnittstellen-ICs und Magnetkoppler zum Einsatz, die diesem Kriterium entsprechen. Vertreter der aktuellen Generation dieser ICs weisen eine Schutzklassifizierung der höchsten Stufe, Klasse 4, der IEC 61000-4-2 auf. Die Datenpins dieser ICs sind gegen ESD mit Amplituden von bis zu  $8kV$  bei direkter Kontaktentladung bzw.  $15kV$  bei Luftentladung vor Beschädigung geschützt.

Ein weiteres Mittel zum Schutz ganzer Schaltungen vor Transienten, stellt die Verwendung von TVS-Dioden zum Überspannungsschutz dar. Parallel zur schützenden Schaltung verbaut, stellen die TVS-Dioden im normalen Betrieb eine sehr hohe Impedanz dar und werden bei Auftreten von Transienten leitend, um den überhöhten Spannungspegel gegen Masse abzuleiten. Eine unidirektionale Ausführung der TVS-Diode verhält sich in Durchlassrichtung geschaltet wie eine gewöhnliche Gleichrichterdiode. In Sperrichtung betrieben, entspricht der Kennlinienverlauf der Charakteristik eines Avalanche-Elements. Abbildung 3.8 zeigt die unidirektionale Charakteristik einer solchen TVS-Diode. Bidirektionale Ausführungen zeigen im positiven und negativen Spannungsbereich der Kennlinie die Charakteristik eines Avalanche-Elements [4].



**Abbildung 3.8:** Kennlinienverlauf einer unidirektionalen TVS-Diode

Die Auswahl eines TVS-Diodentypen zum Schutz elektronischer Komponenten erfolgt nach folgenden vier Kriterien:

- Impulsverlustleistung  $P_{PPM}$
- maximale Sperrspannung  $V_{WM}$
- minimale Durchbruchspannung  $V_{BR}$
- Clampingspannung  $V_C$

$V_{WM}$  ist die maximal zulässige Betriebsspannung, mit der die zu schützende Komponente ordnungsgemäß funktioniert.  $V_{BR}$  wird mit 10 % über der  $V_{WM}$  dimensioniert.  $V_C$  ist der begrenzende Spannungspegel. Alle auftretenden Transienten werden auf diesen Pegel limitiert.  $P_{PPM}$  ergibt sich aus dem Produkt von  $I_{PP}$  eines definierten Stromimpulses und  $V_C$ .

Auch auf Multifunktions-Adapterplatine kommen TVS-Dioden zum Schutz einzelner Komponenten zum Einsatz. Bidirektionale TVS-Dioden schützen das PoE-Modul vor Transienten, die über das Ethernetkabel eingestreut werden können. Transienten, die sich über die USB-Leitung einbringen und ausbreiten, werden durch ein spezielles TVS-Diodenelement begrenzt.

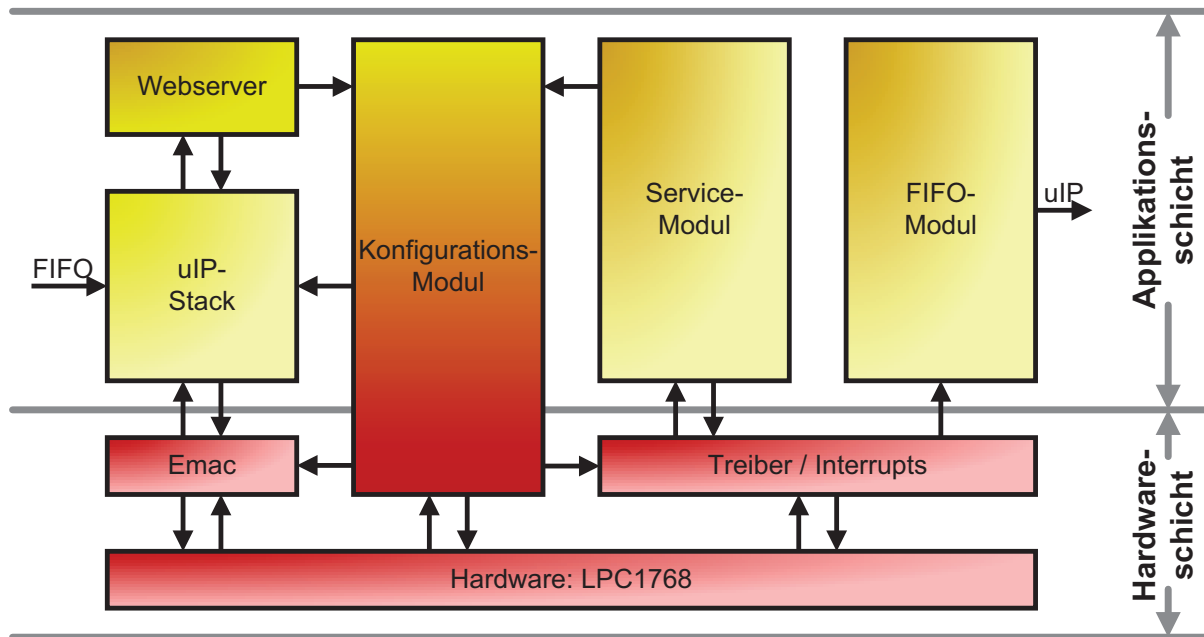
## 3.2 Softwarekonzept

### 3.2.1 Grundkonzept

Da alle physischen Komponenten der Multifunktions-Adapterplatine von einer MCU gesteuert werden, ist selbstverständlich auch ein Softwarepaket notwendig, um die gewünschte Funktionalität zu erreichen. Aufgabe der Software ist es, die verschiedenen physikalischen Schnittstellen zu konfigurieren, zu initialisieren, Daten zu empfangen und in einem definierten Protokoll über die Ethernet-Schnittstelle zu versenden.

Die Programmierung der Software erfolgt in C unter Benutzung der integrierten Entwicklungsumgebung (*engl.*: integrated development environment (IDE)) *LPCEXpresso Version 4* der Firma Code Red Technologies. Diese auf *Eclipse* basierende IDE wird von dem Hersteller kostenfrei zur Verfügung gestellt und verwendet zur Erzeugung des Maschinencodes den *ARM-GCC* Compiler. Das durch Compilieren erzeugte Binärfile mit einer maximalen Größe des Flashspeichers des LPC1768 von 512KiB wird dann einfach auf das *mbed*-Mikrocontrollermodul kopiert und nach Auslösung des RESET ausgeführt.

Um spätere Erweiterungen und nachfolgende Softwarepflege zu vereinfachen, sind alle Programmmodule weitestgehend voneinander abgegrenzt und unabhängig. Die einzelnen Programmteile der *mbed*-Software werden dabei auf zwei Schichten abgebildet. Der schematische Aufbau der *mbed*-Software und die Wechselwirkung der einzelnen Programmmodule untereinander kann der Darstellung 3.9 entnommen werden.

Abbildung 3.9: Modell der *mbed*-Software

Auf unterster Ebene befindet sich die Hardware-schicht und enthält die Programmteile, die unmittelbar auf die Hardware des LPC1768 zugreifen. Diese als Treiber implementierten Programmmodule konfigurieren, aktivieren und deaktivieren die einzelnen Schnittstellen. Bestandteil der Treiber sind ebenso die Interrupt Service Routinen (ISR). Durch ein bestimmtes Ereignis oder durch den Empfang von Daten wird der Programmablauf unterbrochen und die ISR ausgeführt.

Ein zentrales Element der *mbed*-Software stellt das Konfigurationsmodul dar. Dieses Modul nimmt von anderen Programmmodulen Konfigurationsbefehle entgegen und verwaltet die Umgebungsvariablen, die von den Treibern zur Konfiguration der Hardware benötigt werden. Ein direkter Zugriff auf die Hardware der MCU durch das Konfigurationsmodul erfolgt durch das De-initialisieren der Breakout-Sockets, da sie durch die funktionale Mehrfachbelegung keinem festen Treiber zugeordnet sind. Das Konfigurationsmodul hat dabei auch den exklusiven Zugriff auf die optionale Baugruppe des Electrically Erasable Programmable Read-Only Memory (EEPROM). Das Konfigurationsmodul ist daher sowohl der Applikationsschicht als auch der Hardware-schicht zuzuordnen.

Die Programmteile der Hardware-schicht sind speziell an die Hardware des LPC1768 angepasst. Eine Weiterverwendung ist mit gewissen Einschränkungen bestenfalls innerhalb der MCU Familie 17xx möglich. Die Elemente der Applikationsschicht hingegen sind weitestgehend hardwareunabhängig und können so auf anderen Plattformen mit geringen Änderungen wiederverwendet werden.

Um den gesamten Programmablauf der *mbed*-Software so einfach und effizient wie möglich zu gestalten, werden alle Prozesse aus der `main`-Loop in fester Reihenfolge aufgerufen. Erst nach Überprüfung hinsichtlich des Handlungsbedarfs des betroffenen Prozes-

ses wird dieser dann ausgeführt oder übersprungen. Für die Implementierung eines jeden Prozesses gilt, dass die anstehenden Aufgaben hinreichend schnell abgearbeitet werden müssen, damit alle Prozesse innerhalb eines Zeitfensters ausgeführt werden.

### 3.2.2 Hardwareschicht

In dieser Schicht sind all jene Programmteile zusammengefasst, die die Konfiguration, die Initialisierung und das Betreiben der einzelnen physischen Schnittstellen ermöglichen. Zu den in der *mbed*-Software enthaltenen einzelnen Treibern gehören:

- UART-Treiber
- SPI-Treiber
- $I^2C$ -Treiber
- EEPROM-Treiber
- GPIO-Treiber
- CAN-Treiber
- ADC-Treiber
- EMAC-Treiber
- TIMER-Treiber

Der **UART-Treiber** enthält alle Funktionen, die zum Betreiben der Schnittstellentypen RS-232/422/485 notwendig sind. Im Treiber sind die Initialisierungsroutinen für die Peripheriekomponenten **UART1**, **UART2** und **UART3** implementiert, deren physische Funktion über die Breakout-Sockets 3, 4 und 2 realisiert wird. Der Empfang der Daten erfolgt durch Auslösen eines Interrupt der jeweiligen UART-Schnittstelle. Die im Treiber implementierte ISR leitet die empfangenen Daten dann byteweise an einen Zwischenspeicher, der auf dem Prinzip des First In - First Out (FIFO) basiert, weiter. Das Setzen des GPIO zur Aktivierung des DC/DC-Konverters erfolgt ebenso durch diesen Treiber.

Der **SPI-Treiber** stellt die Funktionen zum Betrieb der beiden SPI-Schnittstellen im Master-Mode über die Breakout-Sockets 1 und 3 bereit. Der Betrieb der SPI-Schnittstellen ist als reine 32-Bit Kommunikation konzipiert. Der Treiber enthält dazu Funktionen zum Senden eines 32 Bit langen Kommandowortes. Dieses frei definierbare Datenwort kann entweder als Einzelabfrage oder durch Nutzung eines der beiden im Treiber implementierten Hardwaretimer in einem vorher definierten Intervall gesendet werden. Die über die Leitung des **MISO** empfangene Antwort wird durch Abarbeitung der entsprechenden ISR als 32-Bit Wort an das FIFO-Modul übergeben. Durch die Verwendung des 32 Bit langen Datenrahmens wird die direkte Verwendung von 16-Bit ADCs als SPI-Gerät ermöglicht. Funktionen zum Auslösen des **RESET**-Signals im SPI-Betriebsmodus sind ebenfalls Bestandteil dieses

Treibers. Die Dauer des LOW-Pegels des RESET-Signals lässt sich in Schritten von  $100\mu s$  einstellen.

Der **I<sup>2</sup>C-Treiber** stellt die elementaren Funktionen zur Verfügung, die Breakout-Sockets 2 und 4 mit dem I<sup>2</sup>C-Bus zu verbinden. Jeder der beiden Busteilnehmer kann dabei im Master Transmitter Mode oder im reinen Slave Receiver Mode betrieben werden. Für die Kommunikation über den I<sup>2</sup>C-Bus kann zwischen den beiden Übertragungsgeschwindigkeiten Standard Mode mit  $100kBit/s$  oder Fast Mode mit  $400kBit/s$  gewählt werden. Wie bei SPI ist auch die Auslegung der I<sup>2</sup>C-Schnittstelle sehr stark vom verwendeten Teilnehmer abhängig und muss im Einzelfall speziell angepasst werden. Der Treiber stellt somit lediglich die Funktionen zum Senden eines einzelnen Bytes im Master Mode bzw. für den Empfang eines Bytes im Slave Mode zur Verfügung. Der Slave übergibt das empfangene Byte anschließend an das FIFO-Modul.

Der **EEPROM-Treiber** stellt alle Funktionen zum Betrieb des optionalen EEPROM-Speichermoduls auf dem Breakout-Socket 1 zur Verfügung. Die im Treiber enthaltene Initialisierungsroutine wird dabei bei jedem Start der Multifunktions-Adapterplatine ausgeführt. Damit wird Breakout-Socket 1 bei jedem Start in einem SPI-Modus initialisiert. In der darauf folgenden Phase der Autoerkennung wird das Vorhandensein eines validen EEPROM-Speichers überprüft. Der Ausgang dieser Überprüfung wird dem Konfigurationsmodul gemeldet. Befindet sich auf dem Breakout-Socket 1 der erwartete EEPROM-Speicher mit den Konfigurationsdaten, so verbleibt dieser Socket im Betriebsmodus **FLASH EEPROM**. Führt die Autoerkennung zu einem negativen Ergebnis, so wird Breakout-Socket 1 unverzüglich de-initialisiert. Für das Lesen und Schreiben von Daten enthält der Treiber entsprechende Funktionen, um die übergebenen Variablen der unterschiedlichen Datentypen dauerhaft auf den EEPROM-Speicher abzulegen oder auszulesen.

Der **GPIO-Treiber** ermöglicht es, die Datenleitungen des Breakout-Sockets 1 (Pin 3-7) als digitale Ausgänge zu schalten. Für den Wechsel zwischen den Pegelzuständen **HIGH** und **LOW** sind in diesem Treiber entsprechende Funktionen vorhanden.

Die Treiberfunktionalitäten, die für das Zurücksetzen der bis hier genannten Peripheriekomponenten in den definierten Ausgangszustand verantwortlich sind, werden innerhalb des Konfigurationsmoduls zur Verfügung gestellt.

Der **CAN-Treiber** ermöglicht das Betreiben des CAN2 Controllers des LPC1768. Der Treiber enthält Funktionen zur Initialisierung, De-Initialisierung und dem Empfang von CAN-Nachrichten. Der Empfang von Nachrichten im Standard und im Extended Frame Format werden dabei in gleicher Weise unterstützt. Die Länge der Nutzdatenlast innerhalb einer CAN-Nachricht darf dabei zwischen 0 und 8 Bytes variieren. Der Empfang einer CAN-Nachricht löst auch hier wieder einen Interrupt aus und die im Treiber enthaltene ISR leitet die gesamte Nachricht unverändert an das FIFO-Modul weiter. Der CAN-Treiber bietet die Möglichkeit entweder alle Nachrichten des CAN-Busses (Bypass Mode) zu empfangen oder durch Verwendung des Akzeptanzfilters Nachrichten anhand

der Message ID herauszufiltern. Für den Standard und Extended Frame Modus stehen jeweils zwei Listen zur Definition des Akzeptanzfilters zur Verfügung. Die erste Liste enthält 8 Einträge mit Message IDs der zu empfangenen Nachrichten. Die zweite Liste des Akzeptanzfilters nimmt vier Gruppen IDs auf. Gruppen IDs definieren einen beliebig großen Bereich der zu empfangenen Nachrichten mit entsprechender Message ID. Eine Gruppen ID besteht dabei stets aus einer Start Message ID und einer End Message ID. Jeder Eintrag in den Listen des Akzeptanzfilters lässt sich aktivieren bzw. deaktivieren. Die CAN-Datenübertragungsrate kann frei gewählt werden und ist gemäß der Spezifikation auf maximal  $1\text{MBit/s}$  limitiert.

Der **ADC-Treiber** ermöglicht die Nutzung des in der MCU integrierten 12-Bit A/D-Wandlers. Der Treiber initialisiert und aktiviert 1 bis 4 Kanäle der auf der Multifunktions-Adapterplatine zur Verfügung stehenden analogen Eingänge. Die Abtastrate kann frei gewählt werden und ist auf  $200\text{kHz}$  bei Nutzung nur eines ADC-Kanals limitiert. Die ISR des ADC-Treibers liefert dem FIFO-Modul den entsprechenden Sample innerhalb eines 16-Bit langen Datenwortes. Um jeden Sample eindeutig einen der vier ADC-Kanäle zuordnen zu können, werden die ersten beiden Bits des 16-Bit Wortes binär mit der Kanalnummer versehen. Ein binärer Wert von 00 kennzeichnet den Sample dem Kanal 1 des ADC zugehörig. Das binäre 01 steht für den Zweiten, 10 für den Dritten und ein 11 für den vierten Kanal des ADC. Der Aufbau des 16-Bit Datenwortes kann der Abbildung 3.10 entnommen werden. Funktionen zur Deaktivierung einzelner ADC-Kanäle bzw. für die komplette De-Initialisierung des ADC sind ebenso Bestandteil dieses Treibers.

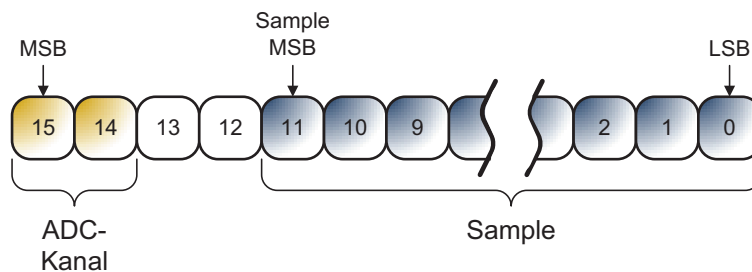


Abbildung 3.10: Aufbau des 16-Bit ADC-Datenwortes

Der **EMAC-Treiber** enthält die Funktionen zur Konfiguration und Initialisierung der Ethernet-Schnittstelle. Neben der Konfiguration des in der MCU integrierten Ethernet-blocks, stellt der Treiber auch die Kommunikation über das Media Independent Interface (MII) sicher. Über dieses Interface erfolgt die Konfiguration des Ethernet-PHY durch den Treiber. Für das Senden und Empfangen der Datenpakete werden durch diesen Treiber ebenso Speicherbereiche des internen SRAM als Puffer reserviert. Auf diese Speicherbereiche wird mittels Direct Memory Access (DMA) zugegriffen, um die Central Processing Unit (CPU) zu entlasten. Der EMAC-Treiber ist Bestandteil einer uIP-Stack Demoapplikation für den LPC1768 und wird ohne Veränderung in das Projekt der *mbed*-Software integriert.

Der **TIMER-Treiber** konfiguriert und initialisiert den Hardwaretimer 0, um Teile des Konfigurations- und Servicemoduls mit Informationen zu Zeitintervallen zu versorgen. Dieser Treiber ist ebenso Taktgeber für die Länge des **RESET**-Impulses für den SPI-Treiber.

### 3.2.3 Das Konfigurationsmodul

Die Konfiguration, Initialisierung und De-Initialisierung der Hardwareschicht geschieht ausschließlich durch das Konfigurationsmodul.

Dazu initialisiert und verwaltet das Konfigurationsmodul eine Vielzahl von globalen Umgebungsvariablen. Diese Variablen bilden die Grundlage für die Treiber, um die entsprechende Peripheriekomponente zu konfigurieren und zu initialisieren. Änderungen an diesen Parametern wirken sich dabei nicht zwangsläufig unmittelbar auf den Betrieb und die Funktion der Multifunktions-Adapterplatine aus. Änderungen an Variablen, die der Konfiguration der Hardware dienen, wie zum Beispiel der Baudrate, werden erst nach Re-Initialisierung der entsprechenden Peripheriekomponente wirksam. Änderungen an anderen Variablen, wie zum Beispiel dem Kommandowort bei aktiver SPI-Schnittstelle, führen zur unmittelbaren Änderung der Funktion. Die globalen Umgebungsvariablen dienen anderen Programmmodulen als Informationsquelle, ob und inwiefern Peripheriekomponenten initialisiert sind oder in welchem Betriebsmodus sich die einzelnen Breakout-Sockets befinden.

Bei einem Start der Multifunktions-Adapterplatine erfolgt durch das Konfigurationsmodul eine Initialisierung der globalen Umgebungsvariablen. Dazu veranlasst das Konfigurationsmodul zunächst die Initialisierung des Breakout-Socket 1 im Betriebsmodus **FLASH EEPROM**. Meldet der Treiber eine geglückte Autoerkennung des EEPROM-Speichers, so werden nun alle globalen Umgebungsvariablen mit den ausgelesenen Werten versehen. Die zum Zeitpunkt der Speicherung aktiven Schnittstellen werden ebenso aktiviert. Wurde dagegen kein valider EEPROM-Speicher erkannt, so werden den globalen Umgebungsvariablen Standardwerte zugewiesen. Somit kann die Multifunktions-Adapterplatine jederzeit in einem definierten Startzustand versetzt werden, um ein gewisses Maß an Grundfunktionalität zu gewährleisten. Das Konfigurationsmodul veranlasst ebenso die Speicherung der Gesamtkonfiguration bei vorhandenem EEPROM-Speicher.

Das Konfigurieren von Teilen der Hardware und *mbed*-Software sowie das Initialisieren und De-Initialisieren von Schnittstellen erfolgt auf Grundlage von Konfigurationsbefehlen. Die vom Webserver oder Service-Modul eingehenden Befehle werden bis zur Abarbeitung durch das Konfigurationsmodul in einer FIFO zwischengespeichert. Ein Konfigurationsbefehl wird stets durch ein Tupel, bestehend aus **Config Code** und **Config Parameter**, repräsentiert. Der ein Byte umfassende **Config Code** bestimmt, welche Aktion bei welchem Teil der Hardware ausgeführt werden soll. Der **Config Parameter**, stets als 32 Bit langes Wort übergeben, bestimmt den neuen Wert, auf den die betroffene Komponente verändert wird. Die Konfigurationsbefehle veranlassen dabei die Änderung der globalen

Umgebungsvariablen innerhalb des Konfigurationsmoduls. Zur Wahrung der Betriebsbereitschaft der Multifunktions-Adapterplatine werden die **Config Parameter** innerhalb des Konfigurationsmoduls auf Gültigkeit überprüft.

Um die Gesamtfunktionalität der Multifunktions-Adapterplatine möglichst flexibel zu gestalten, existiert eine Vielzahl von Parametern zur Konfiguration und damit ebenso viele **Config Codes**. Um Erweiterungen der *mbed*-Software zu erleichtern sind die **Config Codes** in einer Baumstruktur aufgebaut. Die ersten vom MSB ausgehenden 4 Bits des **Config Codes** unterteilen die Multifunktions-Adapterplatine in acht Hardwaregruppen und bilden den **Hardware Group Code**. Die darauf folgenden 4 Bits bilden den **Action Code**. Dieser Code bestimmt, welche Aktion für die betreffende Hardwarekomponente ausgeführt wird. Den Aufbau des **Config Codes** und die Einteilung der **Hardware Group Codes** kann der Abbildung 3.11 entnommen werden. Die gesamte Übersicht aller verfügbaren Konfigurationsbefehlen ist dem Anhang G zu entnehmen.

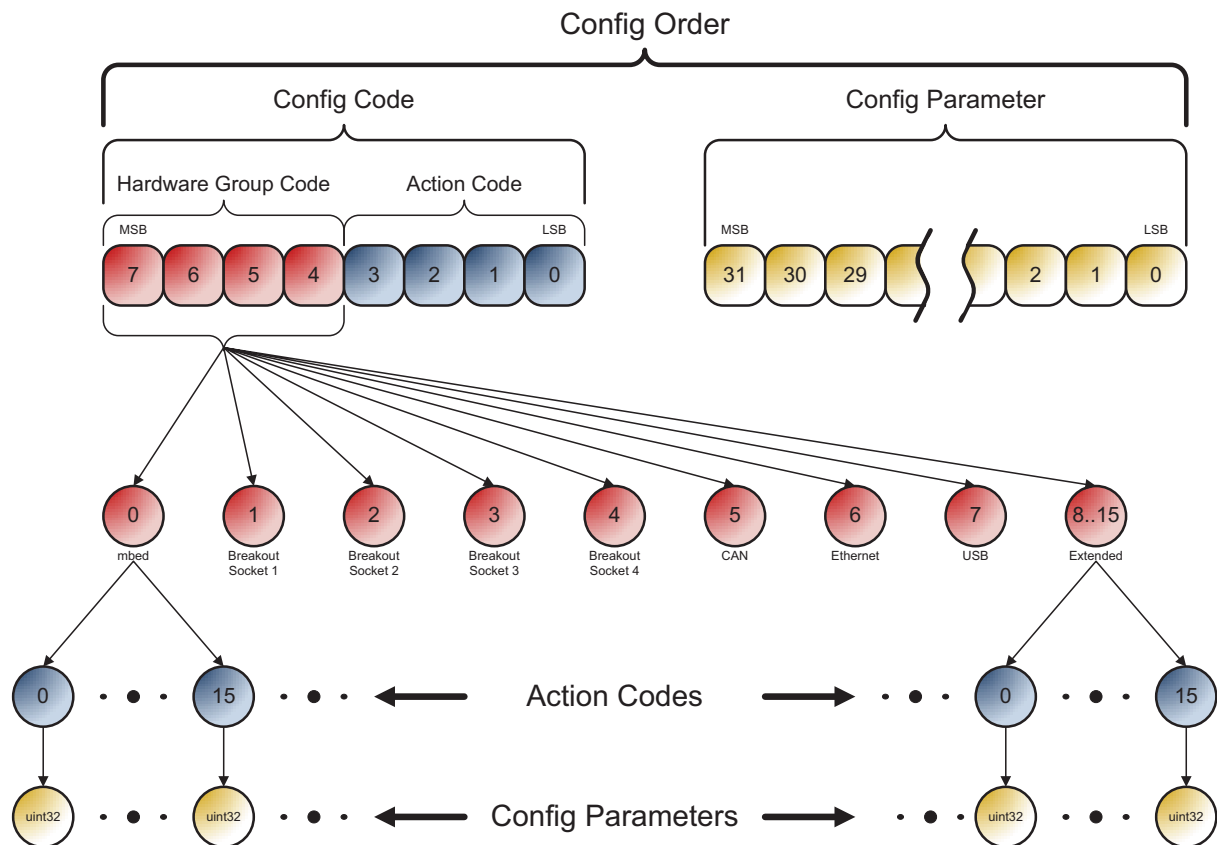


Abbildung 3.11: Aufbau und Struktur der Konfigurationsbefehle

Durch die Teilung des **Config Codes** in zwei 4 Bit lange Codewörter stehen allerdings nur 16 **Action Codes** pro Hardwaregruppe zur Verfügung, welches in einigen Fällen nicht ausreicht. Dazu wurden weitere **Hardware Group Codes** gebildet, die aber keine eigene Hardwaregruppe symbolisieren, sondern lediglich eine Erweiterung darstellen.

Das Konfigurationsmodul übernimmt ebenso die Aufgabe, alle potentiell mit einem DC/DC Wandler ausgestatteten Schnittstellen zeitversetzt zu initialisieren. Dies betrifft

die auf dem Base-Board vorhandene CAN-Schnittstelle und alle vier Breakout-Sockets. Die Initialisierung dieser Schnittstellen erfolgt nach Anfrage im Sekundentakt.

Bedingt durch die mehrfache Funktionalität der Breakout-Sockets, ist eine zentrale De-Initialisierung der betroffenen Datenleitungen unumgänglich. Dies ist ebenso die Aufgabe des Konfigurationsmoduls, indem nach Anfrage jeder Breakout-Socket in einen definierten Ausgangszustand gesetzt wird. Damit wird bei Wechsel eines Breakout-Boards bei laufendem Betrieb der Multifunktions-Adapterplatine eine einheitliche Basis geschaffen, damit der entsprechende Treiber die Schnittstelle erfolgreich initialisiert. Zu diesem Zweck greift das Konfigurationsmodul direkt und unmittelbar auf die Hardware des LPC1768 zu.

### 3.2.4 uIP-Stack und Webserver

Der von Adam Dunkels entwickelte uIP-Stack stellt die Funktionalität bereit, die für eine Kommunikation auf dem Network und Transport Layer des OSI Schichtenmodells über die Ethernet-Schnittstelle notwendig ist. Die hier zum Einsatz kommende Implementierung des uIP-Stacks liegt in der Version 1.0 vor und ist eine spezielle Umsetzung für die LPC17xx Familie. Der Vorteil der Verwendung dieses Stack liegt in seiner geringen Codegröße, den geringen Bedarf an SRAM Speicher und Rechenkapazität. Der uIP-Stack unterstützt dabei folgende Protokolle [14]:

- Internet-Protokoll (IP)
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)
- Internet Control Message Protocol (ICMP)

Der Versand der empfangenen und zwischengespeicherten Daten erfolgt durch Verwendung des UDP. Da es sich bei Verwendung dieses Protokolls um eine verbindungslose Kommunikation handelt, sind sehr hohe Datenübertragungsraten erreichbar. Wie es die Abbildung 3.12 zeigt, ist der Aufbau eines UDP-Paketes sehr einfach gehalten, wodurch eine schnelle Abarbeitung des Datenversands durch die MCU erfolgt. UDP stellt allerdings nur die minimalen Mittel zur ungeschützten und ungesicherten Kommunikation bereit. Durch Verwendung dieses Protokolls wird nicht gewährleistet, dass ein Datenpaket tatsächlich auch sein Ziel erreicht. Der Empfang von UDP-Paketen in der richtigen Reihenfolge wird ebenfalls nicht garantiert. Derartige Fehler zu erkennen und zu beseitigen wird die Aufgabe der Anwendungsschicht sein.

Der auf dem uIP-Stack aufgesetzte Webserver dient als GUI zur Konfiguration und Steuerung der Multifunktions-Adapterplatine. Die zur Interaktion mit den Nutzer notwendigen Seiten werden direkt im Quelltext der *mbed*-Software als dynamische Webseiten gespeichert. Eine Aufgabe des Webserver ist die strukturierte Anzeige der Konfiguration und des Initialisierungszustandes der Komponenten der Multifunktions-Adapterplatine.

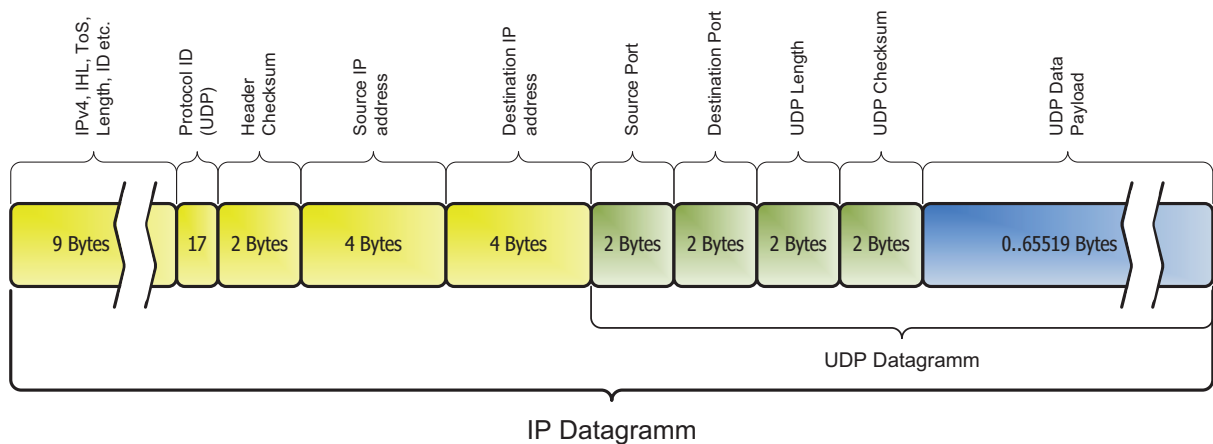


Abbildung 3.12: Aufbau eines UDP-Paketes auf IPv4 Ebene

Die dazu benötigten Daten erhält der Webserver anhand der globalen Umgebungsvariablen von dem Konfigurationsmodul. Die zweite Aufgabe des Webserver ist das Ändern der Konfiguration und das Initialisieren und De-Initialisieren der Schnittstellen durch die GUI. Die auf der Webseite getätigten Eingaben werden durch den Webserver ausgewertet, in Konfigurationsbefehle gewandelt und an das Konfigurationsmodul versendet.

Da der Webserver das Hypertext Transfer Protocol (HTTP) verwendet, wird zur Übertragung der Daten eine TCP/IP Verbindung benötigt. Diese Verbindung wird ebenfalls durch den uIP-Stack bereit gestellt.

Durch die Integration des uIP-Stack und des Webserver in das *mbed*-Softwareprojekt verhält sich die Multifunktions-Adapterplatine wie ein gewöhnlicher World Wide Web (WWW) Teilnehmer im Netz. Für die Konfiguration und Steuerung der Multifunktions-Adapterplatine ist keine spezielle Software und Hardware notwendig. Ein einfacher Browser und Zugang zum Netz, in dem sich die Multifunktions-Adapterplatine befindet, sind hierzu ausreichend.

### 3.2.5 Das FIFO-Modul

Das FIFO-Modul dient dem Zwischenspeichern der empfangenen Daten und dem Aufbau des Datenpaketes, welches als UDP-Paket durch den uIP-Stack versendet wird.

Der Empfang von Daten über die aktiven Schnittstellen löst im entsprechenden Treiber eine ISR aus und übergibt die Daten dem FIFO-Modul. Zum Zweck der Speicherung verwaltet das FIFO-Modul für jede Schnittstellenfunktionalität einen eigenen Speicherbereich. Dieser Speicher puffert den Datenstrom nach dem Prinzip des FIFO und ist als Ringspeicher implementiert. Um die Verweildauer innerhalb einer ISR minimal zu halten, entsprechen die Speicherzellen dem gleichen Datentyp der übergebenen Daten.

Bei einem sehr hohen eingehenden Datenstrom besteht die Gefahr eines Pufferüberlaufs. Durch das Fehlen von freien FIFO-Speicherzellen würden noch nicht versandte Daten

unkontrolliert überschrieben werden. Daher wird vor jeder Speicherung der Zustand des FIFO-Speichers hinsichtlich eines FIFO-Überlaufs hin untersucht. Tritt dieser Zustand ein, so wird der FIFO-Überlauf nach Außen über die Ethernet-Schnittstelle signalisiert. Je nach Konfiguration des entsprechenden FIFO-Speichers werden nun die empfangenen Daten verworfen (blocking mode) oder der älteste im FIFO-Speicher vorhandene Wert wird mit den neuen Daten überschrieben (non-blocking mode).

Befinden sich versandbereite Daten im FIFO-Speicher, so wird durch das FIFO-Modul eine von dem uIP-Stack bereitgestellte Datenstruktur beschrieben, welche die Datennutzlast des UDP-Paketes darstellt. Die ersten beiden Bytes des Datenpaketes enthalten den Wert des **Data Length Counter**. Dieser Wert gibt die Anzahl der nun folgenden Bytes des UDP-Paketes an. Der **Package Counter** versieht das UDP-Datenpaket durch die nun folgenden 2 Bytes mit einer fortlaufenden Paketnummer. Durch die Durchnummerierung der Pakete wird es dem Empfänger möglich, bei Bedarf die Empfangsreihenfolge zu korrigieren. Ebenso kann in einem gewissen Zeitfenster der Verlust von Paketen detektiert werden. Nun werden die im FIFO-Speicher abgelegten Daten in die Datenstruktur übertragen. Den eigentlichen Daten wird dabei stets ein Byte als Identifikator (ID) vorangestellt. Diese durch das Konfigurationsmodul frei wählbare ID symbolisiert eine der acht Hardwaregruppen und spiegelt die Herkunft der Daten wieder. Das nächste Byte gibt die Länge der Daten in Bytes an, die nun aus dem FIFO-Speicher in die Datenstruktur kopiert werden. Anschließend werden die empfangenen Daten hier byteweise abgelegt. Sind in weiteren FIFO-Speichern Daten enthalten, so wiederholt sich der oben beschriebene Vorgang. Liegen keine weiteren versandfertigen Daten im FIFO-Speicher vor oder ist Kapazität der Datenstruktur ausgeschöpft, so ist der Aufbau der UDP-Datennutzlast beendet und das FIFO-Modul meldet dem uIP-Stack die Versandbereitschaft des Paketes. Abbildung 3.13 verdeutlicht anhand eines Beispiels den Aufbau der Nutzlast des UDP-Paketes.

Aufgetretene FIFO-Speicherüberläufe werden signalisiert, indem die entsprechende ID der Hardwaregruppe gefolgt von einem Byte mit der Wertigkeit NULL in das Datenpaket geschrieben wird. Die Datenlänge NULL ist in diesem Fall nicht zulässig und kann vom Empfänger als vorliegender Fehler interpretiert werden. Sofern das Fassungsvermögen der UDP-Datenstruktur es zulässt, wird nun versucht, alle Daten des FIFO-Speichers in das UDP-Paket zu übertragen, um diesen Speicher zu leeren. Dabei können jeweils maximal 256 Bytes transferiert werden.

Da CAN-Nachrichten unterschiedlich lang sein können, wird diesen Daten eine besondere Behandlung beim Beschreiben des UDP-Paketes zuteil. Die ersten beiden Bytes werden hier wie gewohnt geschrieben. Das erste Byte enthält die ID der CAN-Schnittstelle, das zweite Byte die Länge in Bytes der nun folgenden Daten. Jetzt folgt die Message ID des Absenders. Da der Empfang von CAN-Nachrichten sowohl im Standard Frame als auch Extended Frame Format unterstützt wird, kann dieser Eintrag 2 bzw. 4 Bytes umfassen. Um dem Empfänger die Unterscheidung zu ermöglichen, wird das MSB dieses Eintrages kodiert. Eine logische NULL steht dabei für eine Nachricht im Standard Frame Format und die 11 Bit umfassende Message ID ist in dem 2 Byte umfassendem Eintrag abgelegt. Eine

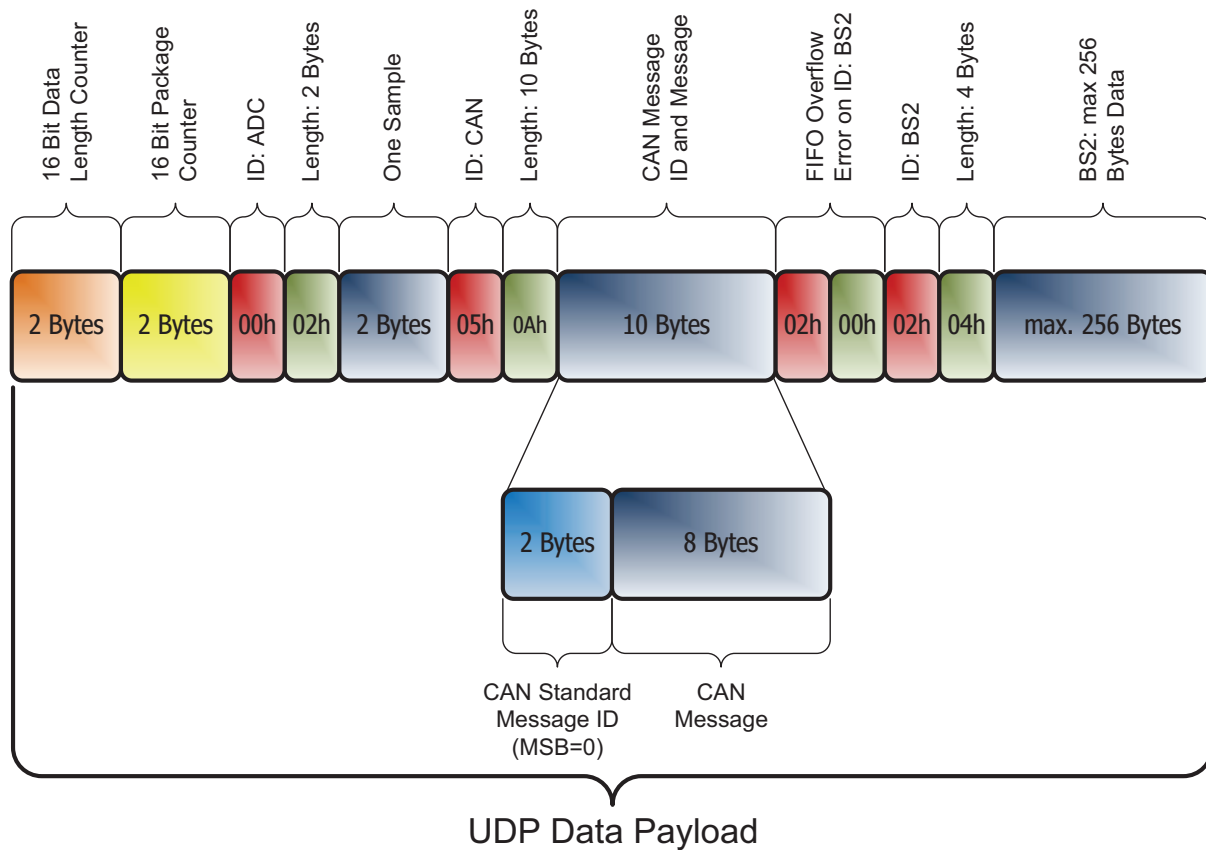


Abbildung 3.13: Exemplarischer Aufbau der UDP-Nutzdaten

logische EINS signalisiert dem Empfänger die Präsenz der CAN-Nachricht im Extended Frame Format. Die 29 Bit lange Message ID ist in einem 4 Byte umfassendem Eintrag abgelegt. Ab hier folgt die Speicherung der Nutzdaten der CAN-Nachricht. Liegen mehrere CAN-Nachrichten im FIFO-Speicher vor, so wiederholt sich der gesamte Vorgang entsprechend.

Die Speicherzelle einer UART FIFO enthält jeweils nur ein einzelnes Byte. Aufgrund der meist sehr geringen Datenübertragungsrate der auf UART basierenden Schnittstellen ist es in vielen Fällen unpraktisch, ein UDP-Paket mit einem einzelnen Byte als Nutzlast zu versenden. Abhilfe schafft hier der durch das Konfigurationsmodul bereitgestellte Parameter **FIFO Character Charge**. Dieser Parameter definiert eine Mindestvorhaltemenge an Zeichen für die drei UART FIFOs. Der Versand erfolgt erst, wenn die Anzahl der im UART FIFO abgelegten Zeichen den Wert dieses Parameters überschreitet. Bleibt der Empfang von Daten über diese Schnittstellen aus, so wird der Inhalt des betroffenen FIFO-Speichers für den Versand freigegeben und geleert. Durch diese Maßnahme lässt sich das Versenden von unnötig vielen UDP-Paketen vermeiden.

### 3.2.6 Das Service-Modul

Das Service-Modul stellt Funktionen bereit, die im Rahmen der Entwicklung und Pflege der *mbed*-Software unerlässlich sind. Dieses Modul besteht aus dem In-Operation Monitor und einem UART-Testteil.

Da die Programmierung des *mbed*-Mikrocontrollermoduls nicht mit spezieller Hardware mit Joint Test Action Group (JTAG) Protokoll erfolgt, ist ein geeignetes Kommunikationsmittel zwischen *mbed*-Hardware und Softwareentwickler zu schaffen. Dabei kann die USB-Schnittstelle des *mbed*-Mikrocontrollermoduls nicht nur zum Übertragen des Binärfiles verwendet werden. Mit entsprechender Initialisierung kann diese Schnittstelle als weitere UART-Schnittstelle genutzt werden. Die Verwendung einer Terminalemulation auf dem PC ermöglicht die Ein- und Ausgabe von Werten in Klartext. Abbildung 3.14 zeigt hier die Verwendung des Programms *HyperTerminal* von Microsoft stellvertretend für eine Terminalemulation. Der In-Operation Monitor initialisiert diese UART-Schnittstelle mit einer Datenübertragungsrate von 57600Bit/s in 8-N-1 Codierung und stellt ein vollständiges Command Line Interface (CLI) zur Verfügung. Die Aufgaben des In-Operation Monitors sind:

- Einsehen von globalen Variablen zur Laufzeit des Systems
- Direkte Eingabe der Konfigurationsbefehle
- Ausführen von UART-Tests

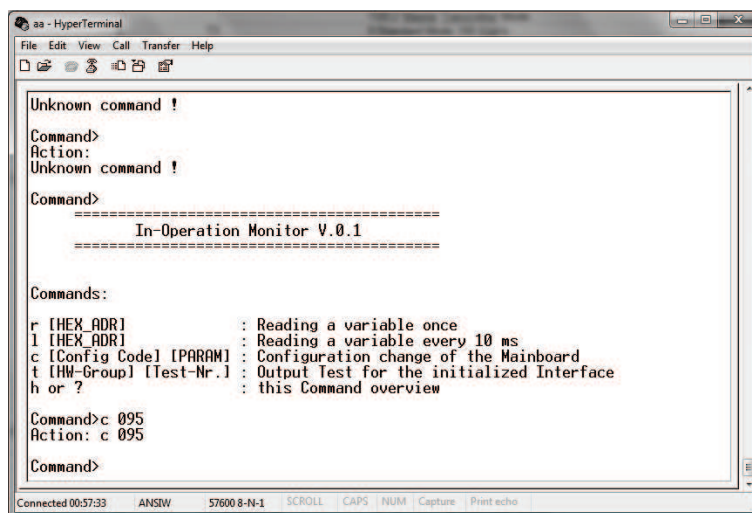


Abbildung 3.14: In-Operation Monitor in der Terminalemulation

Durch Eingabe des Kommandos `r`, gefolgt von einem Leerzeichen und der achtstelligen Speicheradresse in hexadezimaler Schreibweise in Großbuchstaben, wird der aktuelle Wert als vorzeichenlose 32-Bit Zahl im dezimalen System auf der Terminalemulation ausgegeben. Den Variablennamen, die dazugehörige Speicheradresse und die Speichergröße kann dem MAP File entnommen werden. Diese Textdatei wird nach jedem Compilieren des Quelltextes erzeugt und enthält die Speicherbelegung des Zielsystems. Abbildung 3.15

zeigt exemplarisch einen Ausschnitt aus dem MAP File. Für sich rasch ändernde Werte gibt das Kommando 1 bei gleicher Syntax den Wert des Speichers im Takt zu 10 ms aus.

|      |                   |                       |
|------|-------------------|-----------------------|
|      | 0x100010ea        | isSSP1_Test_Once      |
|      | 0x100010ec        | pat_cnt               |
| .bss | 0x100010f0        | 0x14 ./main/time.o    |
|      | 0x100010f0        | sec_counter           |
|      | <b>0x100010f4</b> | <b>ten_ms_counter</b> |
|      | 0x100010f8        | counter_1s            |
|      | 0x100010f9        | counter_10ms          |
|      | 0x100010fa        | isLoopHeader          |
|      |                   | isSSP1_Test_Timer     |
|      | 0x100010fc        |                       |
|      | 0x100010fd        |                       |

Abbildung 3.15: Ausschnitt aus dem MAP File

Durch den In-Operation Monitor lässt sich die Multifunktions-Adapterplatine durch direkte Eingabe der Konfigurationsbefehle auch ohne Verwendung des Webservers konfigurieren. Mit Eingabe des Kommandos **c**, gefolgt von dem dreistelligen **Config Code** und dem **Config Parameter**, beides in dezimaler Form, lassen sich so die gewünschten Aktionen auch bei fehlender Netzwerkverbindung auslösen.

Die Breakout-Sockets 2, 3 und 4 können durch die Eingabe des Kommandos **t** in den Sendebetrieb versetzt werden, sofern sie in einer auf UART basierenden Schnittstellenart initialisiert wurden. Das Testkommando, zudem auch die Angabe des zu stimulierenden Breakout-Sockets und die Auswahl des entsprechenden Tests gehört, aktiviert einen der Testfunktionen im UART Testteil. Eine erneute Eingabe des Kommandos **t** mit der dazugehörigen Nummer des Breakout-Sockets beendet den Test unverzüglich.

| Testfunktion | Test                                                 |
|--------------|------------------------------------------------------|
| 0            | Einmaliges Senden des Zeichens U                     |
| 1            | Periodisches Senden des Zeichen U im 10 ms Intervall |
| 2            | Permanentes Senden des Zeichen U                     |
| 3            | Einmaliges Senden der Testzeichenkette               |
| 4            | Periodisches Senden der Testzeichenkette             |
| 5            | Permanentes Senden der Testzeichenkette              |

Tabelle 3.2: UART Testfunktionen des Servicemoduls

Die im UART Testteil enthaltenen Funktionen dienen zum einen der Verifikation der berechneten Baudraten, die bei einem entsprechendem Test mit dem Oszilloskop oder einem Frequenzmessgerät überprüft werden können. Zum anderen dienen sie dem Nachweis, dass die realisierten Breakout-Boards der RS-232 Schnittstelle sowohl in Senderichtung als auch Empfangsrichtung ordnungsgemäß funktionieren. Die Ausführung eines Tests erfolgt nach Auswahl eines von sechs verfügbaren Einzeltests. Die ersten drei Tests senden dabei das Zeichen U im gewählten Zeitintervall. Die verbleibenden drei Testfunktionen senden eine 64 Byte umfassende Zeichenkette bestehend aus den Ziffern 0-9, gefolgt von kleinen und großen Buchstaben in alphabetischer Reihenfolge.

Abgeschlossen wird die Zeichenkette durch die Steuerzeichen **carriage return** und **line feed**. Der Tabelle 3.2 kann eine Auflistung der sechs verfügbaren Einzeltests pro UART basierender Schnittstelle entnommen werden.

Das Service-Modul mit seinen beiden Komponenten spielt für die Funktionalität der Multifunktions-Adapterplatine keine Rolle und soll von hieran nur beiläufig erwähnt werden. Durch die fehlende Möglichkeit des Debuggen ist dieses Modul aber ein unverzichtbares Werkzeug zur Softwareentwicklung.

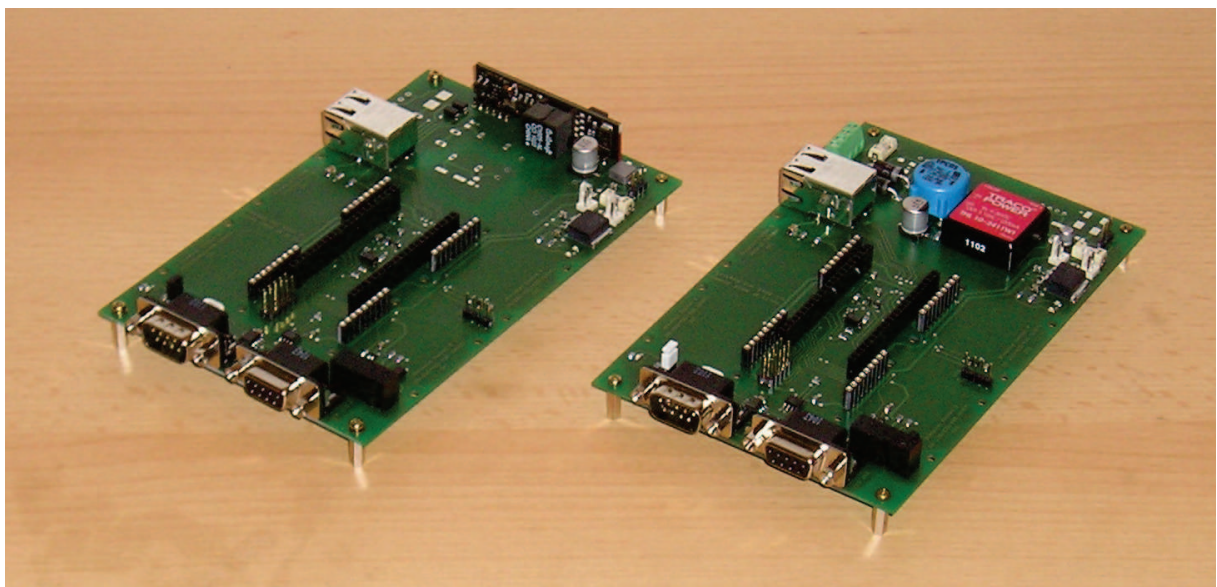
## 4 Hardware-Entwurf und Realisierung

### 4.1 Allgemeines

Im Rahmen der Anfertigung dieser Masterarbeit wurden zwei Base-Boards und drei identische Breakout-Boards mit der Funktionalität der RS-232 Schnittstelle physisch realisiert. Ein weiteres Breakout-Board wurde zur Realisierung eines EEPROM-Speichermoduls angefertigt. Die Breakout-Boards der RS-422 und RS-485 Schnittstelle existieren dagegen als abgeschlossene Konstruktion.

Der Entwurf der Schaltpläne und das Design des Platinenlayouts der Breakout-Boards wurde mit der kostenfreien Version der Software *EAGLE Version 5.11* von CadSoft ausgeführt. Da die Limitierungen der freien Version das Designen von Platinen im Europa-Format nicht zulassen, wurde für das Layout des PCB des Base-Boards die *Professional Version* verwendet.

Um die zur Verfügung stehende Platinenfläche optimal auszunutzen, wurden hauptsächlich elektronische Bauelemente in SMD Bauweise verwendet. Da die filigranen Strukturen der SMD Technik ein hohes Maß an Exaktheit bei der Fertigung der PCBs voraussetzt, wurden alle Platinen, mit Ausnahme des EEPROM-Moduls, von der Firma Beta LAYOUT GmbH gefertigt. Abbildung 4.1 zeigt die beiden Base-Boards im bestückten Zustand.



**Abbildung 4.1:** Die beiden Base-Boards im bestückten Zustand

## 4.2 Aufbau des Base-Boards

In zentraler Position auf dem Base-Board befindet sich das *mbed* Mikrocontroller-Modul. Der Sockel zur Aufnahme des Moduls besteht aus zwei einfachen 20-poligen Buchsenleisten, die dem Rastermaß von  $2,54\text{mm}$  entsprechen.

Für die Realisierung der vier 9-poligen Breakout-Sockets werden qualitativ hochwertige Buchsenleisten der Firma FISCHER ELEKTRONIK verwendet. Die Verwendung dieser Sockel stellt auch bei häufigem Wechsel der Breakout-Boards einen zuverlässigen Kontakt her. Die Kontaktierung zwischen den GPIOs des *mbed* und den Breakout-Sockets kann der Anlage A entnommen werden. Die technischen Details des Base-Board sind in der Anlage B enthalten. Die Abbildung 4.2 zeigt die Platzierung der Grundkomponenten auf dem Base-Board.

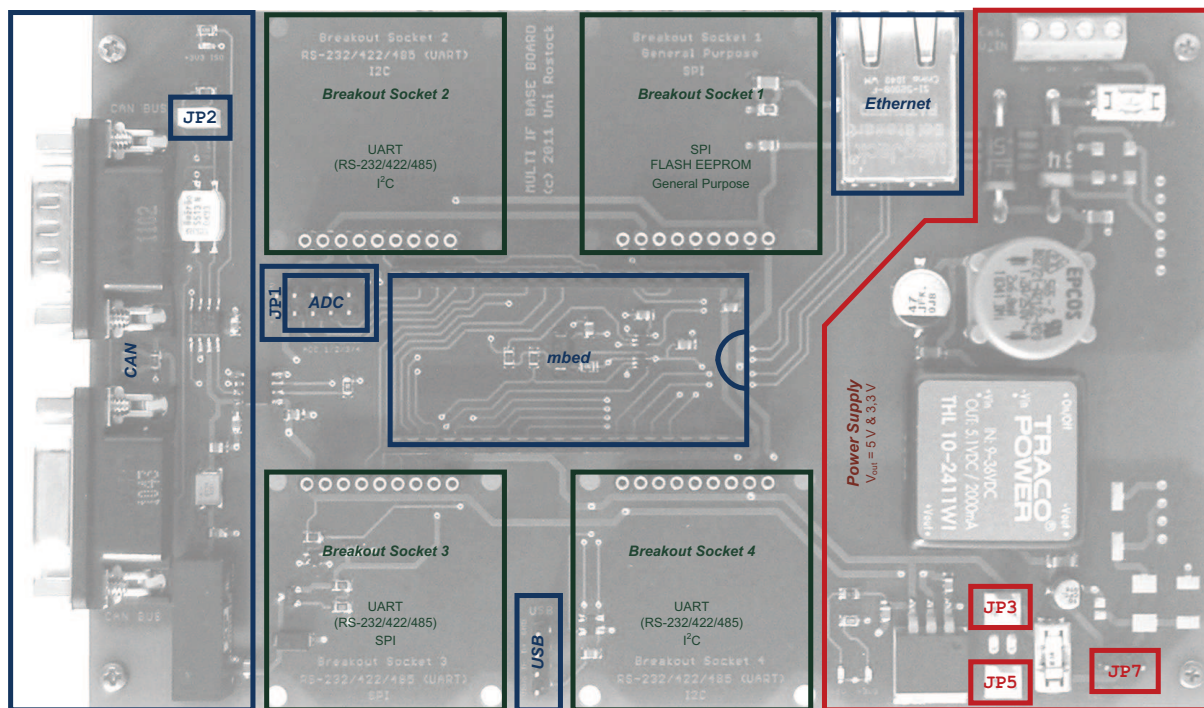
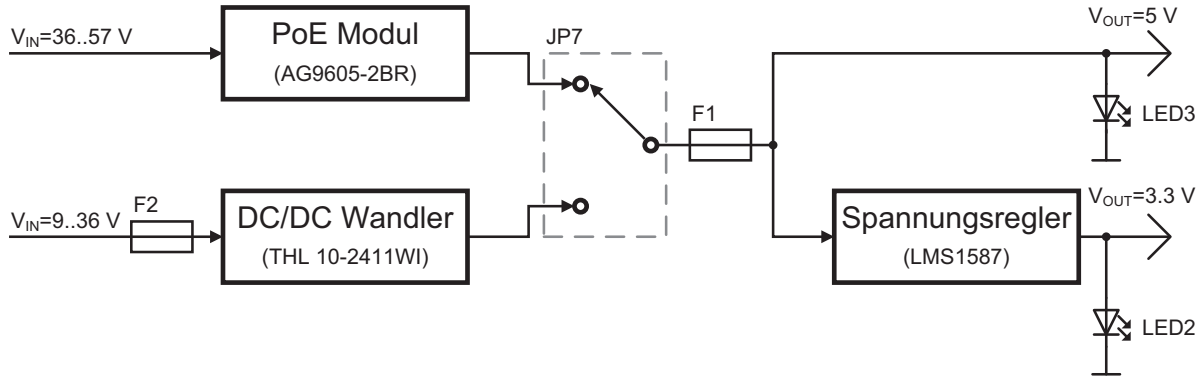


Abbildung 4.2: Komponenten des Base-Boards in der Übersicht

### 4.2.1 Spannungsversorgung

Das Bereitstellen der primären Versorgungsspannung von  $+5\text{V}$  erfolgt entweder durch das PoE-Verfahren oder durch Verwendung eines DC/DC Wandlers. Das PCB Layout des Base-Boards ist so gestaltet, dass beide Verfahren der Spannungsversorgung physisch vorhanden sein können. Die Auswahl der verwendeten Spannungsquelle erfolgt mit JP7. Um die Komponenten der Spannungsversorgung vor Überlastung durch die Verbraucher des Base-Boards zu schützen, wird eine austauschbare, schnell auslösende 2A Sicherung in SMD Technik verwendet. Das Vorhandensein der Versorgungsspannung von  $+5\text{V}$  wird

über die rote LED3 signalisiert. Mit einen Durchflußstrom von  $I_F = 2mA$  ist diese LED besonders energieeffizient.



**Abbildung 4.3:** Schema der dualen Spannungsversorgung des Base-Boards

Die Hauptkomponente des über das PoE-Verfahren bereitgestellten Versorgungsspannung ist das PoE-Modul *AG9605-2BR* der Firma Silvertel. Dieses Modul bezieht die Eingangsspannung direkt aus der Ethernetschnittstelle. Die im Bereich von  $+36V$  bis  $+57V$  anliegende Gleichspannung wird durch einen im Modul integrierten DC/DC Wandler auf eine Ausgangsspannung von  $+5V$  konvertiert. Ein- und Ausgangsspannung sind mit einer Isolationsspannung von  $1500V$  voneinander galvanisch getrennt. Um die Polarität der Eingangsspannung zu detektieren, ist in diesem Modul bereits ein Brückengleichrichter integriert. Das PoE-Modul übernimmt ebenso die Singnaturerkennung und Klassifizierung des PoE-Verfahrens. Das Base-Board verhält sich nach Außen hin wie ein PD der Klasse 0. Um das PoE-Modul vor eingestreuten Spannungsspitzen am Eingang zu schützen, ist den Leitungen VA1/VA2 und VB1/VB2 jeweils eine bidirektionale TVS Diode parallel geschaltet. Wird das PoE-Modul trotz Vorhandenseins nicht zur Bereitstellung der Versorgungsspannung benutzt, so ist der Lastwiderstand R10 unbedingt auf der Platine zu bestücken. Das PoE-Modul setzt eine Minimallast von  $I_{LOAD} = 100mA$  voraus, sonst kann das Modul dauerhaft Schaden nehmen. Der Lastwiderstand R10 mit  $R_{LOAD} = 47\Omega$  ist dabei so gewählt, dass eine Verlustleistung von maximal 1 Watt abgeführt werden kann.

Alternativ dazu kann das Base-Board auch mit Hilfe eines geregelten DC/DC Wandlers versorgt werden. Hauptkomponente dieser Spannungsversorgung ist der *THL 10-2411WI* der Firma Traco Electronic AG. Die Eingangsgleichspannung liegt im Bereich von  $+9V$  bis  $+36V$ . Mit einer Ausgangsspannung von  $U_{OUT} = 5.1V$  kann so eine maximale Last von  $I_{LOAD} = 2A$  versorgt werden. Auch hier ist die Ausgangsspannung mit einer Isolationsspannung von  $1500V$  von der Eingangsspannung galvanisch getrennt. Die als SMD ausgeführte Sicherung F2 begrenzt den Eingangsstrom auf  $0.5A$ . Die stromkompensierte Ringkernndrossel L2 dient hier der Unterdrückung von gleichsinnig auftretenden Störströmen auf den Leitungen der Eingangsspannung.

Durch den linearen Spannungsregler *LMS1587* der Firma National Semiconductor wird

die sekundäre Versorgungsspannung von  $+3.3V$  bereitgestellt. Zur Stabilisierung der Ausgangsspannung sind durch Verwendung dieses Spannungsreglers lediglich die Kondensatoren C4 und C5 mit jeweils  $C = 10\mu F$  notwendig. Das Vorhandensein der sekundären Versorgungsspannung wird durch das Leuchten der gelben LED2 signalisiert. Eine Kühlung des Spannungsreglers ist im normalen Betrieb nicht notwendig. Erst ab einer Lastaufnahme von  $I_{LOAD} = 660mA$  ist der Einsatz einer passiven Kühlung notwendig.

JP3 und JP5 stellen verpolungssichere Steckkontakte zum Abgreifen der primären Versorgungsspannung zur Verfügung. Über JP1 kann ebenfalls jeweils einmal die primäre und sekundäre Versorgungsspannung abgegriffen werden. Abbildung 4.3 zeigt schematisch den Aufbau der dualen Spannungsversorgung.

#### 4.2.2 Schnittstellen

Auf dem Base-Board selbst sind die Schnittstellen für Ethernet, CAN und USB physisch vorhanden. JP1 stellt über eine 2-reihige Stiftleiste den Kontakt zu den vier internen ADC-Kanälen der MCU her.

Die **Ethernet-Schnittstelle** wird auf dem Base-Board durch den Einsatz der Mag45 *SI-52008-F* der Firma Bel Stewart Connector realisiert. Neben dem Abgreifen der Datenleitungen über die Pins 1,3,4 und 5, stellen die Pins 8 bis 11 die spannungsführenden Leitungen für das PoE-Verfahren bereit. Die Datenleitungen sind direkt mit den Kontakten des *mbed*-Sockels verbunden. Durch den im Mag45 integrierten magnetischen Übertrager sind die Datenleitungen zwischen Ein- und Ausgang mit einer Isolationsspannung von  $2000V$  galvanisch voneinander getrennt. Die Anoden der beiden im Mag45 integrierten LEDs, die die Netzwerkaktivität signalisieren sollen, sind mit der Masse verbunden. Diese beiden LEDs werden nicht genutzt, da der Ethernet-PHY vom Typ *DP83848J* Bestandteil des *mbed*- Mikrocontrollermoduls ist und die für die Ansteuerung der LEDs notwendigen Leitungen nicht direkt nach außen geführt werden. Das Abschirmblech des Mag45 Gehäuses ist über eine Parallelschaltung aus R6 und C10 mit der Masse verbunden. Diese Shield-Schaltung dient der Eliminierung von Transienten, die über das Gehäuse eingestreut werden.

Die **CAN-Schnittstelle** untergliedert sich in die zwei Hauptkomponenten Spannungsversorgung und Datenkomponente. Die kontrolliert schaltbare Spannungsversorgung ermöglicht, mit Hilfe der galvanischen Trennung, die Datenkomponente mit einer isolierten Spannung zu versorgen. Die Datenkomponente ermöglicht den vom Base-Board entkoppelten Datenverkehr über den CAN-Bus.

Der geregelte DC/DC Wandler vom Typ *IF0503S* der Firma XP Power wandelt die  $+5V$  Primärspannung des Base-Boards in die isolierte Versorgungsspannung von  $+3.3V$ . L3 und C9 am Ausgang des DC/DC Wandlers bilden hierbei ein LC-Filter, um die Restwelligkeit der Ausgangsspannung zu minimieren. Das Vorhandensein der isolierten

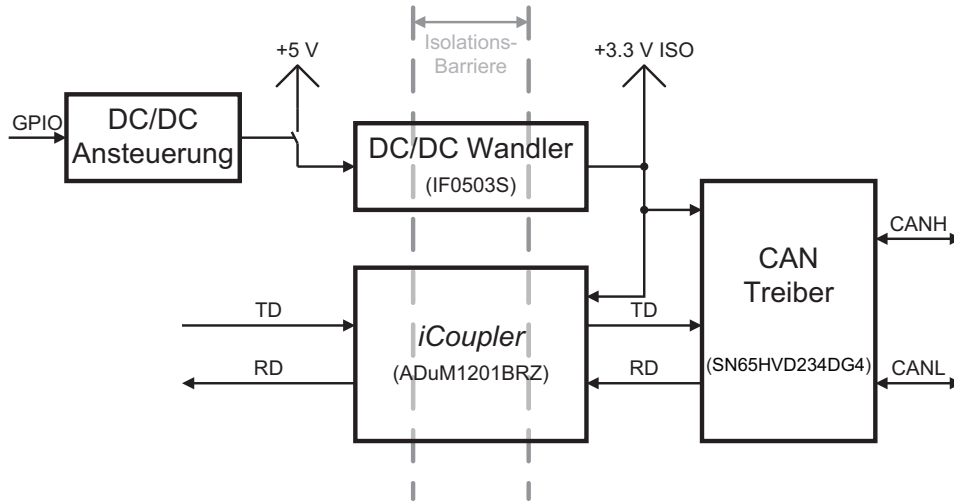
Spannung von  $+3.3V$  wird hier durch das Leuchten der stromsparenden gelben LED1 signalisiert.

Das Zu- bzw. Abschalten der CAN-Spannungsversorgung erfolgt durch einen selbst-sperrenden (Anreicherungstyp) p-Kanal MOS-FET vom Typ *IRLM6401*. Bei Anliegen des Massepotentials am Gate schaltet der MOS-FET vollständig durch und über Source und Drain wird die primäre Versorgungsspannung für den DC/DC Wandler durchgeschaltet. Der Spannungspegel am Gate wird dabei durch einen GPIO des *mbed* bestimmt. Zwischen *mbed* und MOS-FET befindet sich zwecks der Pegelanpassung IC7 vom Typ *74HCT1G125GW*, welcher die Funktionalität eines Tri-State-Gatters hat. Der Dateneingang des Gatters ist mit der Masse verbunden. Das Schalten des Ausgangspegels erfolgt über den Eingang **OUTPUT ENABLE** des Tri-State-Gatters. Nach dem Einschalten des *mbed*-Mikrocontrollermoduls sind alle GPIOs als digitale Eingänge geschaltet und stellen so einen hochohmigen Widerstand dar. Der Eingang **OUTPUT ENABLE** wird daher über einen Pull-Up Widerstand auf ein definiertes Potential gebracht und der Ausgang des Tri-State-Gatters ist im hochohmigen Zustand. Damit beträgt  $V_{GS} = 0V$  und der MOS-FET bleibt nach dem Einschalten des Systems gesperrt. Durch Deklaration des steuernden GPIO als digitalen Ausgang und dem Setzen des **LOW** Pegels, schaltet das Tri-State-Gatter den Eingang durch und der **LOW** Pegel liegt nun auch am Ausgang des Gatters und dem Gate des MOS-FETs an. Mit  $V_{GS} = -5V$  schaltet der MOSFET durch und versorgt den DC/DC Wandler mit der primären Versorgungsspannung. Um die Spannungsversorgung für die CAN-Schnittstelle zu deaktivieren, wird der GPIO in den Zustand **HIGH** versetzt.

Für die galvanische Trennung der Datenleitungen der CAN-Schnittstelle kommt der magnetische Übertrager *ADuM1201BRZ* der Firma Analog Devices zum Einsatz. Da die CAN-Schnittstelle für Datenübertragungsraten von bis zu  $1MBit/s$  konzipiert ist, wird hier eine Version des *iCoupler* der mittleren Geschwindigkeitsklasse verwendet. Dieser Übertrager lässt Datenübertragungsraten von bis zu  $10MBit/s$  zu und wird auf beiden Seiten der Isolationsbarriere mit einer Spannung  $V_{DD} = 3.3V$  versorgt. Für die Datenleitungen des Logikpegels **RD** und **TD** verfügt der *iCoupler* über jeweils einen Übertragungskanal in die entsprechende Richtung.

Der IC3 vom Typ *SN65HVD234DG4* der Firma Texas Instruments übernimmt als Treiber die Wandlung des Logikpegels in den Leitungspegel. Die Ringkerndrossel L1 mit  $L = 51\mu H$  unterdrückt die in die Leitungen des CAN-Bus eingestreuten symmetrischen und asymmetrischen Störimpulse. Über JP2 kann der Terminierungswiderstand R2 mit  $R_{TERM} = 120\Omega$  aktiviert oder deaktiviert werden. Eine geschlossene Jumperstellung schließt den CAN-Bus mit diesem Parallelwiderstand ab und lässt die CAN-Schnittstelle als Endteilnehmer agieren. Zur physischen Verbindung der CAN-Schnittstelle mit dem Bus steht hier jeweils eine als Buchse und eine als Stecker ausgeführte 9-polige D-SUB Verbindung zur Verfügung. Die Kontakte für die Signale **CANH** und **CANL** sind jeweils miteinander verbunden. So kann die CAN-Schnittstelle mittels StICKKABEL durch Nutzung einer beliebigen der beiden D-SUB Verbindungen mit dem Bus verbunden werden. Zur Ausbildung einer Daisy Chain Bustopologie stehen beide D-SUB Verbindungen zur

Verfügung. Die Kontaktbelegung kann der Tabelle B.2 entnommen werden. Die Gehäuse der D-SUB Stecker/Buchse sind über eine gemeinsame Shield-Schaltung mit dem isolierten Massepotential verbunden. Abbildung 4.4 zeigt eine schematische Darstellung der CAN-Schnittstelle.



**Abbildung 4.4:** Schematische Darstellung der CAN-Schnittstelle

Die **USB Schnittstelle** ist als 4-polige Stiftleiste im Rastermaß  $2,54\text{mm}$  physisch realisiert. Um die Eingänge des *mbed*-Mikrocontrollermoduls vor auftretenden Transienten zu schützen, sind die Datenleitungen D+ und D- über ein TVS Diodenarray abgesichert. Das hier verwendete Bauelement vom Typ *PRTR5V0U2X* ist speziell für den Schutz der USB Datenleitungen konzipiert.

Die Erzeugung des **RESET**-Signals für die beiden als **SPI Schnittstelle** nutzbaren Breakout-Sockets 1 und 3 erfolgt durch zwei GPIOs des *mbed*-Mikrocontrollermoduls. Diese GPIOs bilden jeweils ein Eingangssignal eines AND Gatter vom Typ *74LVC1G08GW*. Zur Wahrung eines definierten Pegels werden die GPIOs durch Pull-Up Widerstände auf das Potential der sekundären Versorgungsspannung gebracht. Der andere Eingang eines jeden AND Gatters entspricht dabei stets dem HIGH Pegel. Somit nimmt das **RESET**-Signal unmittelbar nach Einschalten des Systems den Zustand HIGH an. Das Auslösen des **RESET**-Zustandes erfolgt dann logikgleich mit dem Setzen des GPIO auf den Zustand LOW.

Jeder auf dem Base-Board zum Einsatz kommende IC ist in direkter Nähe der Versorgungsspannung  $V_{DD}$  mit einem Stützkondensator beschaltet. Bei auftretenden Spannungsschwankungen und Stromspitzen sichern diese Kondensatoren kurzfristig die ordnungsgemäße Spannungsversorgung der ICs. Für diese Aufgabe werden ausschließlich SMD-Keramik-Vielschicht-Kondensatoren mit  $C = 100\text{nF}$  verwendet.

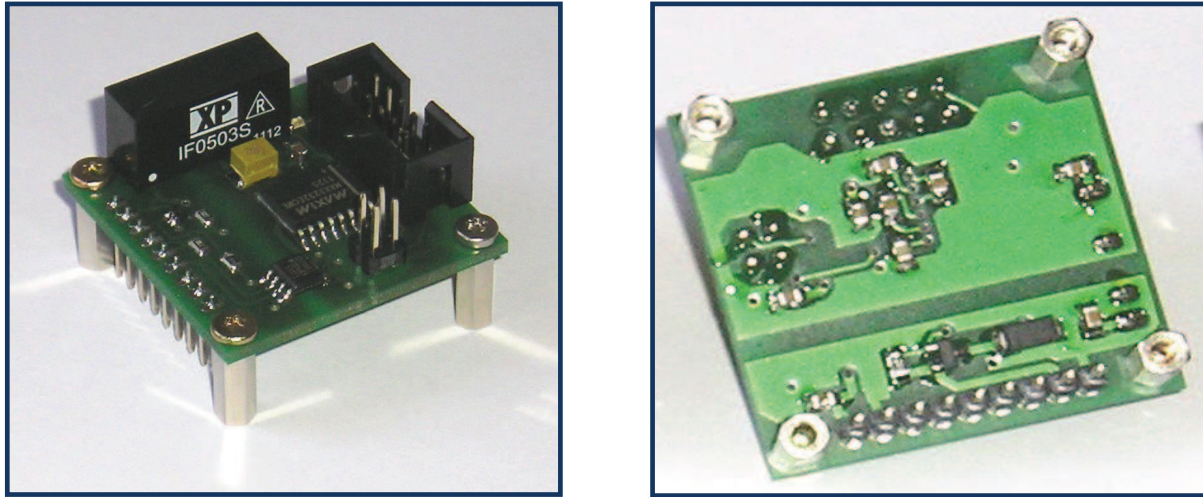
### 4.3 Aufbau der Breakout-Boards

Der Aufbau der Breakout-Boards zur Realisierung der Schnittstellentypen RS-232/422/485 ist in der Grundstruktur identisch zum Aufbau der CAN-Schnittstelle. All diesen Breakout-Boards gemein ist das Vorhandensein der Ansteuerung zum Zu- bzw. Abschalten des DC/DC Wandlers zur Bereitstellung der isolierten Versorgungsspannung von  $V_{DD} = 3,3V$ . Beschaltung und Bauelemente sind identisch zur Spannungsversorgung der CAN-Schnittstelle und das Vorhandensein der isolierten Spannung wird auch hier durch Leuchten einer gelben LED signalisiert. Zur Überwindung der Isolationsbarriere der Datenleitungen wird hier ebenfalls jeweils ein magnetischer Übertrager der *ADuM*-Serie von Analog Devices verwendet. Alle elektronischen Bauelemente werden auf der einheitlichen Fläche von  $35,56mm \times 35,56mm$  untergebracht. Eine 9-polige Stiftleiste stellt dabei jeweils den Kontakt zwischen Breakout-Board und Breakout-Socket her. Vier Bohrungen mit dem Durchmesser von  $2,8mm$  erlauben ein Verschrauben des Breakout-Boards mit dem Base-Board.

Wesentliche Unterschiede zwischen den Breakout-Boards der Schnittstellen RS-232/422/485 bestehen dagegen in der Verwendung unterschiedlicher Treiber-ICs und der physischen Kontaktierung der abgehenden Datenleitungen. Auf diese Unterschiede wird in den folgenden Unterkapiteln näher eingegangen. Das Breakout-Board zur Realisierung des EEPROM-Speichers unterscheidet sich dagegen grundlegend, da die zweckgebundene Aufgabe einem völlig anderem Konzept folgt.

#### 4.3.1 Aufbau des RS-232 Breakout-Boards

Das RS-232 Breakout-Board nutzt zur entkoppelten Übertragung der Datenleitungen den IC *ADuM1201ARZ*. Der Treiber-IC *MAX3232ECWE+* der Firma MAXIM wandelt den Logikpegel des UART in den Leitungsspannungspegel. Als besonders energieeffizientes Bauelement der neuesten Generation verwendet dieser IC eine Versorgungsspannung von  $V_{DD} = 3,3V$ . Die zur Erzeugung des RS-232 Pegels notwendigen Ladungspumpen werden hier durch vier SMD-Keramik-Vielschicht-Kondensatoren mit jeweils  $C = 100nF$  realisiert. Der Treiber-IC enthält zwei separate Transceiver, wobei nur einer auch tatsächlich verwendet wird. Um den Treiber-IC durch unzulässige Spannungspegel vor Beschädigung zu schützen, ist der Eingang des zweiten Transceivers **T2IN** mit der Masse verbunden. Der so am Ausgang **T2OUT** anliegende positive RS-232 Pegel wird durch Kurzschluss an den Eingang **R2IN** weitergeleitet. Der Treiber-IC ist für eine maximale Datenübertragungsrate von  $250kBit/s$  konzipiert.



**Abbildung 4.5:** Realisiertes Breakout-Board der RS-232 Schnittstelle

Die physische Kontaktierung zur RS-232 Datenleitung erfolgt durch einen 10-poligen Wannenstecker. Ein entsprechend konfektioniertes Flachbandkabel mit D-SUB Steckverbindung erlaubt die Verwendung der üblichen RS-232 Leitungen mit 9-poligen D-SUB Verbindungen an beiden Enden. Um die Kommunikation über NULL-Modem als auch über 1:1 kontaktierte Kabel gleichermaßen zu ermöglichen, erlaubt JP1 das Vertauschen der Signale TX und RX.

Die Belegung der Kontakte des Wannensteckers für beide Stellungen von JP1 kann der Tabelle C.2 entnommen werden. Weitere technische Details zum RS-232 Breakout-Board bietet die Anlage C. Abbildung 4.5 zeigt eines der drei gefertigten und in Betrieb genommenen Breakout-Boards der RS-232 Schnittstelle.

#### 4.3.2 Aufbau des RS-422 Breakout-Boards

Die Konstruktion des RS-422 Breakout-Boards sieht ebenso die Verwendung des *ADuM1201ARZ* als magnetischen Übertrager vor. Werden an die Übertragungsgeschwindigkeit der RS-422 Schnittstelle höhere Ansprüche gestellt, so kann auch ein Übertrager einer höheren Geschwindigkeitsklasse verwendet werden. Der *MAX3490ECSA+* der Firma MAXIM übernimmt hier die Rolle des Treiber-IC und gewährleistet die Full-Duplex Kommunikation mit einer Übertragungsgeschwindigkeit von bis zu  $12\text{MBit/s}$  innerhalb eines RS-422 Bussystems. Der Eingangswiderstand dieses Treiber-IC entspricht mit  $R_{IN} = 12\text{k}\Omega$  genau 1 UL für die RS-485 bzw.  $\frac{1}{3}$  UL für die RS-422 Spezifikation. Fail-Safe Schaltung, als auch Strombegrenzung und thermische Abschaltung, sind bereits in diesen Treiber-IC integriert.

Die Verbindung zum RS-422 Bus erfolgt durch eine 6-polige Printklemme. Eine Shield-Schaltung ist für die Verbindung mit dem Kontakt 6 vorgesehen. Der Tabelle D.2 kann die Belegung der Printklemme entnommen werden. Der Terminierungswiderstand mit

$R_{TERM} = 100\Omega$  kann über JP1 aktiviert bzw. deaktiviert werden. Detaillierte technische Daten enthält die Anlage D.

#### 4.3.3 Aufbau des RS-485 Breakout-Boards

Als Halb-Duplex Schnittstelle konzipiert, verfügt das RS-485 Breakout-Board über drei Datenleitungen auf der Ebene des Logikpegels. Die zusätzliche als **Ready to Send (RTS)** bezeichnete Steuerleitung schaltet den Treiber-IC zwischen reinem Sendebetrieb und Empfangsbetrieb um. Daher ist hier der Einsatz des *ADuM1301ARW* als Entkoppler der drei Datenleitungen vorgesehen. Der als Treiber-IC vorgesehene *MAX3485ECSA+* besitzt die gleichen Eigenschaften und Ausstattungsmerkmale des auch schon bei der RS-422 Schnittstelle vorgesehenen Treiber-IC. Der Logikpegel **HIGH** der Steuerleitung **RTS** versetzt den Treiber-IC dabei in den Sendezustand. Nimmt das **RTS** Signal dagegen den Zustand **LOW** an, so agiert der Treiber-IC als reiner Empfänger.

Die Verbindung zum RS-485 Bus wird über eine 4-polige Printklemme hergestellt. Auch hier steht ein Kontakt zur Verbindung mit einer Shield-Schaltung zur Verfügung. Die Kontaktbelegung der Printklemme kann der Tabelle E.2 entnommen werden. Durch JP1 kann auch hier der Terminierungswiderstand mit  $R_{TERM} = 120\Omega$  zu- bzw. abgeschaltet werden. Detaillierte technische Informationen sind der Anlage E zu entnehmen.

#### 4.3.4 Aufbau des EEPROM Breakout-Boards

Als peripheres Endgerät konzipiert, ist der Aufbau des EEPROM Breakout-Boards völlig verschieden von den zuvor vorgestellten Breakout-Boards. Da die einzig verwendeten elektronischen Komponenten hier aus dem IC des EEPROM-Speichers und einem Stützkondensator bestehen, erfolgt der Aufbau auf einer Lochrasterplatine. Der hier zum Einsatz kommende EEPROM-Speicher IC *25LC256* der Firma MICROCHIP ist in der Lage, *32KiB* Daten durch Ansteuerung über die SPI Schnittstelle zu speichern. Der Einsatz des optionalen EEPROM-Speichers ist ausschließlich für das Breakout-Socket 1 vorgesehen. Technische Details können der Anlage F entnommen werden.

## 5 Implementierung der mbed-Software

### 5.1 Aufbau des Softwareprojektes und Hauptprogrammablauf

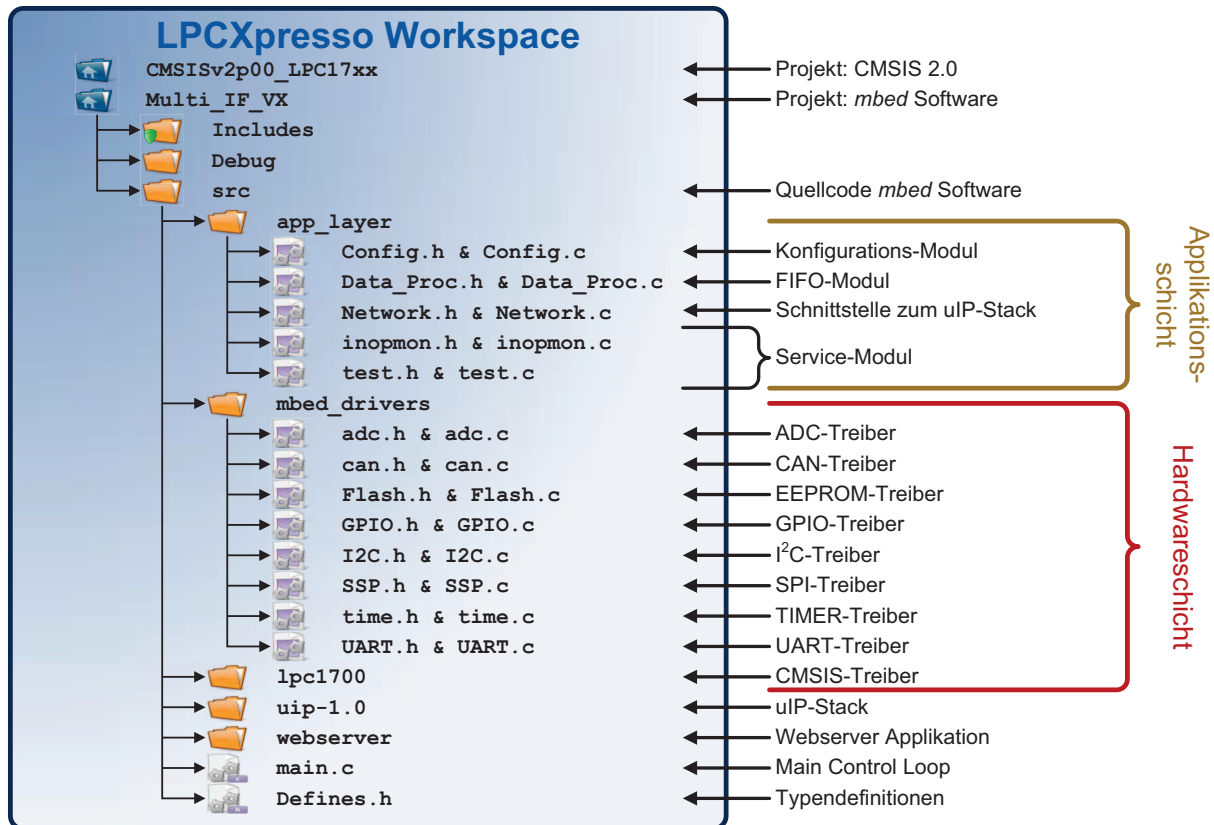


Abbildung 5.1: Projektstruktur der mbed-Software

Die Projektübersicht der IDE *LPCEXpresso* enthält zur Implementierung der *mbed*-Software zwei Softwareprojekte. Das eigentliche *mbed*-Softwareprojekt *Multi\_IF\_VX*, das *X* im Projektnamen dient als Platzhalter für die fortlaufende Versionsnummer, steht in Abhängigkeit zum Projekt *CMSIS\_v2p00\_LPC17xx*. Die Gesamtstruktur der *mbed*-Software ist in Abbildung 5.1 dargestellt.

Der Cortex Microcontroller Software Interface Standard (CMSIS) definiert ein auf den Cortex-M basierendes Mikrocontroller-System. Das hier verwendete CMSIS Softwareprojekt liegt in der Version 2.0 vor und stellt Namen- und Typdeklarationen für den Zugriff auf die Register des LPC1768 bereit.

Der Ordner *Includes* enthält Pfadangaben, damit das *mbed*-Softwareprojekt auf externe Inhalte zurückgreifen kann. Das Einbinden erfolgt durch die IDE über die Projektein-

stellung. Das Binärfile und das **MAP File** werden durch das Compilieren des Softwareprojektes erzeugt und befinden sich danach im Ordner **Debug**.

Der gesamte Quellcode des *mbed*-Softwareprojektes ist im Ordner **src** abgelegt. Der Ordner **app\_layer** enthält dabei die Programmmodule, die der Applikationsschicht zugeordnet werden. Im Ordner **mbed\_driver** sind all jene Programmteile hinterlegt, die die Hardwareschicht repräsentieren. Der Ordner **lp1700** enthält teils modifizierte Bestandteile der CMSIS Treiber-Bibliothek. Unter diesen Treibern, die für den Betrieb des uIP-Stack notwendig sind, befindet sich auch der EMAC-Treiber. Dieser Ordner enthält weiterhin einen Programmteil, der als Start Routine das *mbed*-Mikrocontrollermodul initialisiert und die Kontrolle des weiteren Programmablaufs an die *mbed*-Software übergibt. Der uIP-Stack ist im Ordner **uip-1.0** enthalten. Im Ordner **webserver** sind alle Programmteile des Webserver und die dynamischen Webseiten hinterlegt. Die im Wurzelverzeichnis des Quelltextordners befindliche Datei **Defines.h** enthält die Typendefinition des Datentypes **boolean** und Definitionen einiger häufig verwendeter GPIOs.

Die Quelldatei **main.c** enthält mit der Funktion **main()** den Einsprungpunkt und den Start der *mbed*-Software. Aus der Hauptfunktion heraus werden zunächst die Initialisierungsroutinen der einzelnen Programmmodule aufgerufen und ausgeführt. Danach ist die Multifunktions-Adapterplatine betriebsbereit und der weitere Programmablauf erfolgt innerhalb der Main-Control-Loop, die als nie endende **while(1)** Schleife implementiert ist. In fester Reihenfolge werden nun die Control-Loops der einzelnen Programmmodule aufgerufen.

## 5.2 Implementierung der Hardwareschicht

### 5.2.1 Der ADC-Treiber

Zur Wandlung analoger Signale in zeitdiskrete Werte greift der ADC-Treiber auf zwei Hardwarekomponenten des LP1768 zurück. Der Hardware **Timer1** wird für die Einhaltung des vorgegebenen Abtastintervalls verwendet. Der in der MCU integrierte ADC übernimmt dann nach Anforderung die Wandlung des analogen Signals.

Beide Hardwarekomponenten werden durch die im Treiber enthaltene Funktion **ADC\_Init()** initialisiert. In dieser Phase erfolgt die Berechnung der Auslösefrequenz  $f_{TIMER1}$  der ISR des **Timer1**, die sich nach der Anzahl der ausgewählten ADC-Kanäle  $n$  und der angegebenen Abtastfrequenz  $f_t$  nach der Formel 5.1 ergibt.

$$f_{TIMER1} = n * f_t \quad (5.1)$$

Der Wert von  $f_{TIMER1}$  wird dabei auf die maximale Abtastfrequenz von  $200kHz$  limitiert. Nach erfolgter Initialisierung startet die A/D-Wandlung.

Mit der Frequenz  $f_{TIMER1}$  wird der Interrupt des **Timer1** ausgelöst und die ISR schreibt die Anforderung einer Wandlung für den an der Reihe anstehenden Kanal in das ADC Kontrollregister. Das Ende einer Wandlung löst den Interrupt des ADC aus und die ISR liest den Wert aus dem entsprechenden Datenregister aus. Innerhalb der ISR erfolgt die Codierung des ADC Ursprungs kanals im Sample gemäß Kapitel 3.2.2. Dieser 16-Bit Wert wird dann über einen Funktionsaufruf an das FIFO-Modul übergeben.

Die Funktion `ADC_unInit()` beendet die Aktivität der A/D-Wandlung unmittelbar und versetzt den ADC und den **Timer1** in den uninitialisierten Ausgangszustand.

### 5.2.2 Der CAN-Treiber

Die Initialisierung des CAN2 Controllers erfolgt durch Aufruf der in diesem Treiber enthaltenen Funktion `CAN_Init()`. Der GPIO p2.0 der MCU wird zur Aktivierung der CAN-Schnittstelle des Base-Boards als digitaler Ausgang mit dem Zustand LOW deklariert. Hier erfolgt auch die Berechnung der Werte für das Bus-Timing auf Grundlage der gewählten Baudrate durch Aufruf der Funktion `calc_BTR()`.

Das Setzen der CAN-Filter erfolgt durch das Konfigurationsmodul durch Aufruf der Treiberfunktion `CAN_SetACCF()`. Änderungen an der Filtereinstellungen können auch bei initialisierter Schnittstelle vorgenommen werden und wirken sich umgehend auf die CAN-Funktionalität aus. Sieht die Konfiguration die Verwendung des Akzeptanzfilters vor, so wird dieser Filter entsprechend den Einstellungen konfiguriert.

Der Akzeptanzfilter stellt eine in Hardware realisierte Look-up Table dar, die allen CAN-Controllern der MCU gemeinsam zur Verfügung steht. Zum Beschreiben der Look-up Table mit den Daten der Filterlisten stehen 2 KiB gesonderter RAM-Speicher zur Verfügung. Dieser RAM wird in 32 Bit umfassende Speicherzellen verwaltet. Die Filterlisten für das CAN Standard Frame Format werden vollständig in diesen Speicher übertragen, wobei jede RAM-Speicherzelle zwei Listeneinträge aufnimmt. Neben den zu filternden Message IDs, sind in den Speicherzellen Bitpositionen zur Angabe des betreffenden CAN-Controller und zum Aktivieren/Deaktivieren einzelner Filter vorgesehen. Das Ablegen eines Filtereintrages im CAN Extended Frame Format belegt dabei eine Speicherzelle des RAM des Akzeptanzfilters. Da in diesen Einträgen keine Bitposition zur Aktivierung/Deaktivierung eines Filtereintrages vorhanden ist, werden aus den Filterlisten nur jene Einträge übernommen, die auch entsprechend der Konfiguration aktiv sind. Damit die Einträge des Akzeptanzfilters auch wirksam sind, müssen die einzelnen Message IDs in aufsteigender Reihenfolge gespeichert werden. Die Funktion `sort_Filter()` sortiert alle Einträge der Filterlisten des Konfigurationsmoduls mittels des Bubble Sort Verfahrens. Den Aufbau der im Treiber verwendeten Look-up Table des Akzeptanzfilters soll Abbildung 5.2 verdeutlichen.

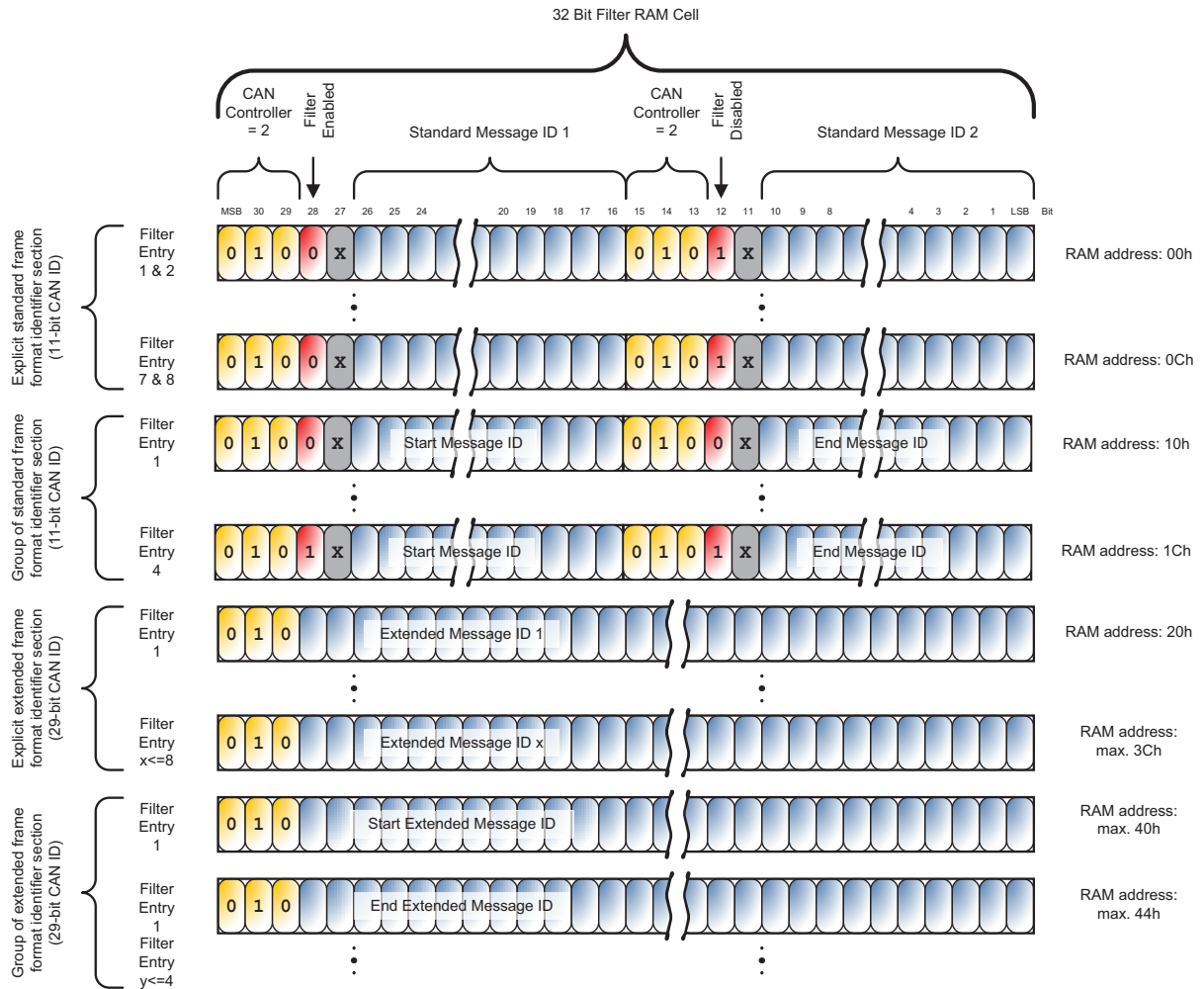


Abbildung 5.2: Speicherbelegung des Akzeptanzfilter RAM

Der Empfang einer CAN-Nachricht löst den Interrupt aus und die ISR überträgt die Daten des Frame Registers, die Message ID des Absenders und die beiden 32 Bit breiten Datenregister in eine Datenstruktur, die in `can.h` definiert wird. Diese Struktur wird dann an das FIFO-Modul übergeben.

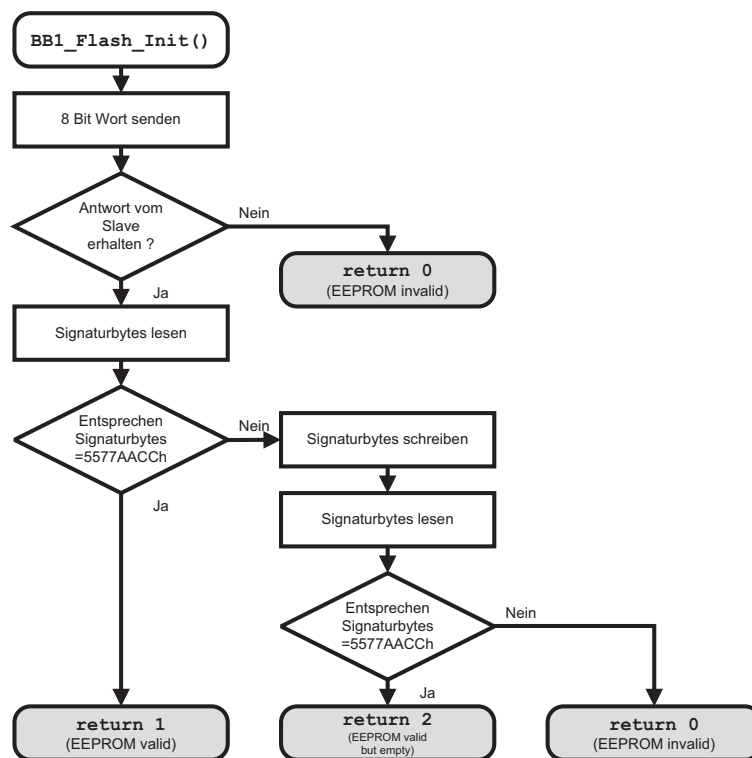
Zur Deaktivierung der CAN-Schnittstelle enthält der Treiber die Funktion `CAN_unInit()`. Durch die Deklaration des GPIO `p2.0` als digitalen Eingang wird die Spannungsversorgung der Schnittstelle abgeschaltet.

### 5.2.3 Der EEPROM-Treiber

Die Initialisierung des EEPROM-Speichers auf dem Breakout-Socket 1 wird durch Aufruf der Funktion `BB1_Flash_Init()` durchgeführt. Dadurch wird dieser Breakout-Socket mit fester  $SCLK = 100kHz$  im SPI-Mode 3 initialisiert. Danach erfolgt die Überprüfung der Validität des EEPROM-Speichers.

Die im Treiber enthaltene Funktion `check_Flash_present()` führt diesen Teil der Autoerkennung durch. Im ersten Schritt wird dazu ein beliebiges Byte als Kommandowort an

den potentiellen Slave versandt. Durch permanentes Abfragen des SPI Empfangs-FIFO wird innerhalb eines Zeitintervalls auf eine Antwort gewartet. Bleibt eine Antwort aus, so kann davon ausgegangen werden, dass der Breakout-Socket 1 gar nicht bzw. mit keinem SPI Slave bestückt ist. Das Konfigurationsmodul wird durch den Rückgabewert 0 darüber informiert und veranlasst umgehend die De-Initialisierung des Breakout-Socket 1. Verlieft der erste Test positiv, so werden nun die vier Signaturbytes ausgelesen und mit dem Sollwert von 5577AACCh verglichen. Bei Übereinstimmung ist die Validität des EEPROM-Speichers hinreichend bewiesen. Der Rückgabewert 1 teilt dem Konfigurationsmodul mit, dass der EEPROM-Speicher mit einem kompletten Datensatz der globalen Umgebungsvariablen vorhanden ist. Sind die Signaturbytes dagegen nicht an den Speicheradressen 00h bis 03h vorhanden, so versucht die Treiberfunktion die Signaturbytes an diese Adressen zu schreiben. Ein erneutes Auslesen und Vergleichen der Signaturbytes gibt letztlich Gewissheit über die Präsenz eines EEPROM-Speichers ohne Datensatz. Verlieft dieser letzte Test erfolgreich, so wird dem Konfigurationsmodul dies durch den Rückgabewert 2 mitgeteilt und der Breakout-Socket 1 behält seine Initialisierung bei. Ein missglückter letzter Versuch wird hingegen mit dem Rückgabewert 0 quittiert. Den Ablauf dieser Autoerkennung kann der Abbildung 5.3 entnommen werden.



**Abbildung 5.3:** Ablauf der Autoerkennung eines validen EEPROM-Speichers

Für das Lesen und Speichern von Daten stellt der EEPROM-Treiber ein Application Programming Interface (API) zur Verfügung. Mit dem Aufruf der Funktion `Flash_Read_Start()` wird dem Treiber der Beginn eines Lesevorganges signalisiert. Das Lesen der Daten erfolgt dann mit Hilfe der Funktion `Flash_Read()`. Der Aufruf der Funktion erfolgt mit der Länge des Datentyps in Byte als Übergabeparameter. Die ausgelesenen

Datenbytes sind dann in dem vorzeichenlosen 32-Bit Rückgabewert dieser Funktion enthalten. Nun folgen entweder weitere Lesebefehle oder der Lesevorgang wird mit Aufruf der Funktion `Flash_Read_End()` beendet.

Das Speichern von Variablen folgt dem gleichen Prinzip. Ein Speichervorgang wird stets mit Aufruf der Funktion `Flash_Write_Start()` eingeleitet. Mit dem Funktionsaufruf `Flash_Write()` wird die übergebene Variable gespeichert. Damit der Treiber über die Anzahl der zu schreibenden Bytes informiert ist, enthält ein zweiter Übergabeparameter die Wortlänge der Variablen in Bytes. Einem Schreibbefehl können weitere folgen. Das Ende des Schreibvorgangs wird durch Aufruf der Funktion `Flash_Write_End()` signalisiert.

Das Verwalten der aktuellen Lese- bzw. Schreibadresse übernimmt der EEPROM-Treiber. Jeder abgeschlossene Lese-/Schreibvorgang setzt den Adresszeiger auf `04h` zurück. Die Partitionierung des EEPROM-Speichers in 64 Byte große Speicherbereiche macht eine Unterbrechung des Speicherns bei Erreichen dieser Grenze notwendig. Nach dem Schreiben des letzten Bytes eines Speicherbereiches führt der Speicher IC selbständig einen ca. *5ms* dauernden Schreibzyklus durch. Innerhalb dieser Phase bleiben Schreib- und Leseversuche wirkungslos. Auch dieser Umstand wird vom Treiber berücksichtigt und bleibt nach außen hin unsichtbar.

Die einzelnen Aufrufe zum Lesen und Schreiben von Variablen müssen direkt aufeinanderfolgen. Vergeht zwischen zwei Lese- bzw. Schreibbefehlen zu viel Zeit, so geht die SPI Signalleitung CS wieder in den Ruhezustand zurück. Nachfolgende Lese-/Schreibbefehle werden dann nicht mehr ausgeführt. Diesem Umstand trägt auch die Wahl des sehr geringen SCLK Rechnung.

#### 5.2.4 Der $I^2C$ -Treiber

Die im  $I^2C$ -Treiber enthaltenen Funktionen `I2C1_Init()` und `I2C2_Init()` initialisieren jeweils das entsprechende  $I^2C$ -Interface der MCU und stellen diese Funktionalität auf den Breakout-Sockets 2 und 4 zur Verfügung. Entsprechend der Konfiguration kann jede Schnittstelle im reinen Master Transmitter Mode oder im Slave Receiver Mode betrieben werden.

Das Senden eines einzelnen Bytes erfolgt durch den Aufruf der Funktion `I2C_Engine()`. Als Übergabeparameter dient die Portnummer, die die zu sendende  $I^2C$ -Schnittstelle identifiziert. Der Wert 1 steht dabei für Schnittstelle `I2C1` auf Breakout-Socket 2, der Wert 2 aktiviert Schnittstelle `I2C2` auf Breakout-Socket 2. Die entsprechende Schnittstelle muss dazu als Master konfiguriert und initialisiert sein.

Die Kommunikation zwischen Master und Slave läuft dabei innerhalb einer Zustandsmaschine ab, die innerhalb der ISR implementiert ist. Jeder Zustandswechsel löst einen Interrupt aus und anhand des  $I^2C$  Status-Codes erfolgt die entsprechende Aktivität.

Gestartet wird die Zustandsmaschine, indem der Master eine Startkondition über den  $I^2C$ -Bus initiiert. Bei fehlerfreier Kommunikation werden dann die in der Abbildung 5.4 aufgeführten Zustände durchlaufen. Das vom Slave empfangene Datenbyte wird an das FIFO-Modul weitergegeben.

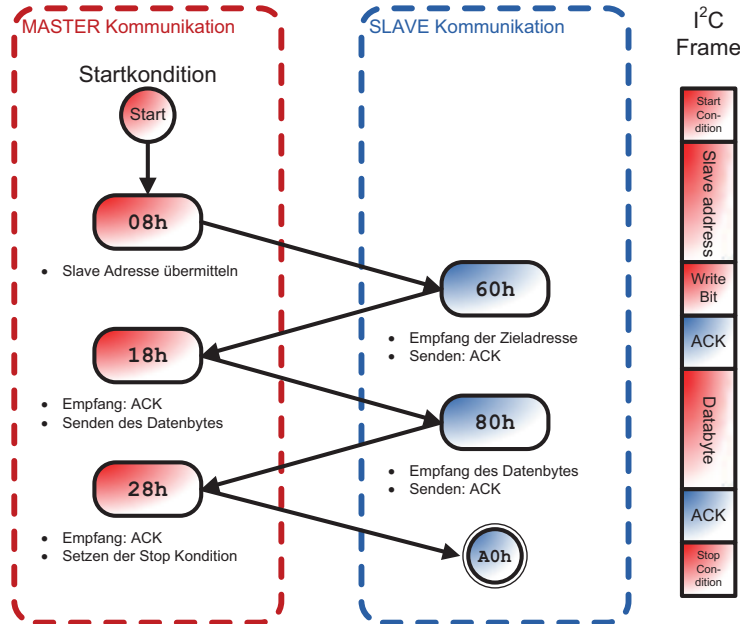


Abbildung 5.4:  $I^2C$ -Kommunikation zwischen Master und Slave als Zustandsmaschine

### 5.2.5 Der SPI-Treiber

Durch Aufruf der Treiberfunktionen `SSP0_Init()` und `SSP1_Init()` werden die beiden SPI fähigen Synchronous Serial Ports (SSP) des LPC1768 initialisiert. `SSP0` stellt die SPI Funktionalität für Breakout-Socket 3 und `SSP1` für Breakout-Socket 1 zur Verfügung.

Sieht die Konfiguration eine Abfragefrequenz für den SSP größer NULL vor, so wird ein Hardwaretimer initialisiert. Für `SSP0` wird der `TIMER2`, für `SSP1` der `TIMER3` verwendet. Um eine Vollaustastung der Multifunktions-Adapterplatine durch den Betrieb der SPI-Schnittstellen zu verhindern, wird die Abfragefrequenz dahingehend limitiert, dass die Busauslastung pro SSP kleiner 50% beträgt. Die maximale Abfragefrequenz  $f_{tMAX}$  steht in direkter Abhängigkeit zum konfigurierten `SCLK` gemäß der Formel 5.2.

$$f_{tMAX} = \frac{SCLK}{66} \quad (5.2)$$

Bei konfigurierter Abfragefrequenz wird im entsprechenden Zeitintervall die ISR des Hardwaretimers ausgelöst. Das 32 Bit umfassende Kommandowort wird nun byteweise in den Sendepuffer des SSP geschrieben. Der Empfang eines Datenbytes über die Leitung des MISO löst den Empfangsinterrupt aus und die ISR speichert diesen Wert zwischen.

Nach vier empfangen Bytes ist die Antwort des Slaves komplett und das 32 Bit Datenwort wird an das FIFO-Modul übergeben.

Mit Hilfe der im Treiber enthaltenen Funktionen `Single_Req_SSP0()` und `Single_Req_SSP1()` können Einzelabfragen durchgeführt werden. Diese Funktionen werden auf Anfrage aufgerufen, wenn die Abfragefrequenz mit NULL konfiguriert wurde.

Für die Erzeugung des RESET-Signals stehen die Funktionen `Reset_SSP0()` und `Reset_SSP1()` zur Verfügung. Bei initialisierten SSP veranlassen diese Funktionen das Schalten des PIN 3 der entsprechenden Breakout-Sockets auf den Zustand LOW. Die Funktionen `SSP0_Reset_Hold()` und `SSP1_Reset_Hold()` werden im Zyklus von  $100\mu s$  vom TIMER-Treiber aufgerufen und überprüfen den Ablauf des Zeitintervalls zum Halten des RESET-Zustandes. Nach Ablauf des konfigurierten Zeitintervalls nimmt der entsprechende GPIO durch Wirken dieser Funktionen wieder den Zustand HIGH an.

### 5.2.6 Der UART-Treiber

Der UART-Treiber initialisiert die Schnittstellen `UART1`, `UART2` und `UART3` auf den Breakout-Sockets 3, 4 und 2. Die Initialisierung erfolgt durch Aufruf der Funktionen `UART1_Init()`, `UART2_Init()` und `UART3_Init()`. Um die Spannungsversorgung durch den DC/DC Wandler für die Breakout-Boards herzustellen, wird die Steuerleitung auf Pin 7 des Breakout-Sockets als digitaler Ausgang mit dem Zustand LOW deklariert. Für die im Betriebsmodus RS-485 befindlichen Breakout-Sockets wird zusätzlich die RTS-Signalleitung auf Pin 6 als digitaler Ausgang mit LOW-Pegel deklariert. Der Zustand von Pin 6 versetzt damit den Transceiver der RS-485 Schnittstelle in den Empfangsmodus.

Die zur Initialisierung benötigte Funktion `UART_Devidev_Calc()` berechnet die korrekten Registerwerte, die zur Erzeugung der Baudrate durch den in der MCU integrierten Baudratengenerator nötig sind.

Bei Empfang eines jeden Zeichens über eine Schnittstelle des UART wird ein Interrupt ausgelöst und die entsprechende ISR liest das Datenbyte aus dem Empfangsregister aus. Dieses Byte wird dann unverzüglich dem FIFO-Modul übergeben.

### 5.2.7 Der GPIO- und TIMER-Treiber

Der Aufruf der im **GPIO-Treiber** enthaltenen Funktion `GPIO_Init()` versetzt das Breakout-Socket 1 in den GPIO Betriebsmodus. Dazu werden alle GPIOs, die mit den Pins 3-7 des Breakout-Socket verbunden sind, als digitale Ausgänge mit den Zustand dem LOW deklariert.

Der Wechsel des Pegelzustandes auf HIGH erfolgt durch Aufruf der Funktion `set_GPIO()`. Der ganzzahlige Übergabeparameter entspricht der Nummer des Pins des

Breakout-Sockets. Wird diese Funktion mit dem Übergabeparameter 1 aufgerufen, so nehmen alle steuerbaren Leitungen den Zustand HIGH an.

Die Funktion `clr_GPIO()` versetzt den als Übergabeparameter angegeben Pin in den Zustand LOW. Ein Aufruf mit dem Übergabeparameter 0 versetzt alle 5 Pins in den Zustand LOW zurück.

Der **TIMER-Treiber** enthält die Funktion `Timer_Init()` zur Initialisierung des Hardwaretimers `TIMER0`. Nach Aufruf dieser Funktion wird im Intervall von  $100\mu s$  der Interrupt ausgelöst und innerhalb der ISR werden entsprechende Zählerstände um `EINS` inkrementiert.

Die als Prozess periodisch aufgerufene Funktion `Timer_Control_Loop()` überprüft die Zählerstände und erzeugt so weitere Zeitintervalle von der Dauer  $10ms$  und  $1s$ . Nach Ablauf eines  $10ms$  Intervalls werden im Bedarfsfall Funktionen des Servicemoduls aufgerufen. Im Zeitintervall von  $1s$  wird durch Aufruf einer Funktion des Konfigurationsmoduls überprüft, ob eine Anfrage zur zeitversetzten Initialisierung einer Schnittstelle vorliegt.

Die Funktionen `Start_SSP0_Reset_Timer()` und `Start_SSP1_Reset_Timer()` starten bzw. beenden die Zeitsteuerung zum Halten des RESET-Zustandes der als SPI initialisierten Schnittstellen.

## 5.3 Implementierung der Applikationsschicht

### 5.3.1 Das Konfigurationsmodul

Mit Aufruf der Funktion `Config_Init()` durch den Programmteil `main.c` wird das Konfigurationsmodul initialisiert und in einen definierten Ausgangszustand gebracht. Neben der Initialisierung einiger Variablen erfolgt durch diese Funktion auch die Deklaration der entsprechenden GPIOs als digitalen Ausgang zur Ansteuerung der vier *mbed*-LEDs. Mit Aufruf der Funktion `Flash_Start()` aus der Initialisierungsroutine heraus wird entschieden, mit welchem Datensatz der globalen Umgebungsvariablen die Multifunktions-Adapterplatine gestartet wird. Liefert der EEPROM-Treiber im Zuge der Autoerkennung einen Rückgabewert von 1, so wird der Datensatz von dem validen EEPROM-Speicher geladen. Ein davon abweichender Rückgabewert veranlasst das Konfigurationsmodul durch Aufruf der Funktion `setDefault_Configuration()` die globalen Umgebungsvariablen mit Standardwerten zu initialisieren. Der Funktionsaufruf erfolgt mit einem booleschen Ausdruck der angibt, ob der Funktionsaufruf zur Erstinitialisierung (`TRUE`) oder im laufenden Betrieb (`FALSE`) stattfindet. Diese Unterscheidung ist notwendig, da zur Veränderung aller globalen Umgebungsvariablen im laufenden Betrieb die Multifunktions-Adapterplatine im Vorfeld vollständig de-initialisiert wird. Nach Setzen der Standardkonfiguration ist lediglich die Ethernet-Schnittstelle und Breakout-Socket 1 aktiv, sofern ein valider EEPROM-Speicher detektiert wurde.

Das Laden eines Datensatzes der globalen Umgebungsvariablen geschieht durch Aufruf der Funktion `Load_Flash_Configuration()`. Auch hier gibt der boolesche Übergabeparameter an, ob der Aufruf zum Zeitpunkt der Erstinitialisierung oder im laufenden Betrieb vorgenommen wird. Im zweiten Fall wird die Multifunktions-Adapterplatine vollständig de-initialisiert. Mit Hilfe der API des EEPROM-Treibers werden nun die Daten vom EEPROM-Speicher eingelesen und den globalen Umgebungsvariablen zugewiesen. Abgespeicherte Initialisierungszustände der einzelnen Schnittstellen werden dabei ausgewertet und die Multifunktions-Adapterplatine wird dementsprechend initialisiert und ist danach einsatzbereit.

Mit Aufruf der Funktion `Save_Flash_Configuration()` lässt sich ein kompletter Datensatz an globalen Umgebungsvariablen auf dem EEPROM-Speicher schreiben. Die Übergabe der Daten an den EEPROM-Treiber erfolgt ebenfalls mit der bereitgestellten API des Treibers. Da der fortlaufende Schreibvorgang auf keinen Fall unterbrochen werden darf, werden zu Beginn der Routine alle aktiven Interruptleitungen deaktiviert. Nach Beendigung des Schreibvorganges werden diese Interruptleitungen wieder aktiviert.

Die Funktion `Config_Loop()` wird periodisch durch die Main-Control-Loop aufgerufen und überprüft das Vorhandensein von abzuarbeitenden Konfigurationsbefehlen. Diese Befehle werden durch Aufruf der Funktion `cfg_cmd_input()` dem Konfigurationsmodul übergeben und in einer eigener FIFO mit 32 Speicherzellen abgelegt. Dieser Funktion werden als Übergabeparameter der `Config Code` und der `Config Parameter` übermittelt. Die Umsetzung der in der FIFO gespeicherten Konfigurationsbefehle erfolgt durch Aufruf der Funktion `set_Config_Param()`. Entsprechend des `Hardware Group Code` und des `Action Code` werden hier nun Umgebungsvariablen mit den Werten des `Config Parameter` versehen und Schnittstellen de-initialisiert oder initialisiert. Die `Config Parameter` werden dabei stets auf Gültigkeit hin überprüft. Ungültige Konfigurationsbefehle werden ignoriert.

Die Funktion `SlowMo_Startup()` ist für die zeitversetzte Initialisierung der Breakout-Sockets 1-4 und der CAN-Schnittstelle verantwortlich. Diese Funktion wird im Intervall von 1s durch den TIMER-Treiber aufgerufen und erteilt die Freigabe zur Initialisierung einzelner Schnittstellen.

Das zentrale De-Initialisieren der Breakout-Sockets erfolgt durch Aufruf der Funktion `bbx_unInit()`, mit `x=1, 2, 3` oder `4` für den entsprechenden Breakout-Socket.

### 5.3.2 Das FIFO-Modul

Eine der Hauptaufgaben des FIFO-Moduls ist die Initialisierung und Verwaltung der verschiedenen, nach dem Prinzip des FIFO funktionierenden, Zwischenspeicher. Für jede einzelne Schnittstellenfunktionalität existiert ein eigener FIFO-Speicher. Jeder dieser Speicher besitzt einen Lese- und einen Schreibzeiger. Weiterhin verfügt jeder FIFO-Speicher über zwei weitere boolesche Ausdrücke. Das `RTS Flag` nimmt den Zustand `TRUE` an, sofern

versandbereite Daten im Speicher hinterlegt sind. Das **Error Flag** signalisiert hingegen ein Überlaufen des FIFO-Speichers. Der Aufbau eines solchen FIFO-Speichers kann der Abbildung 5.5 entnommen werden.

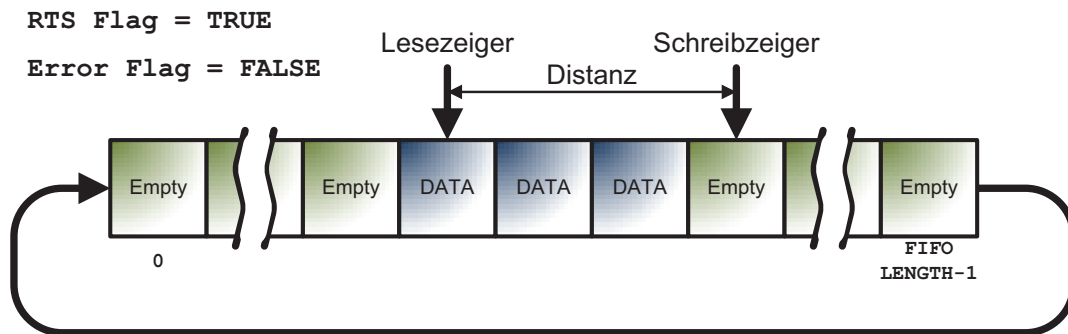


Abbildung 5.5: Aufbau und Struktur eines FIFO-Speichers

Die Funktionen `get_Data_IF()` ( $IF = \{UART1, UART2, UART3, ADC, SSP0, SSP1, CAN, I2C1, I2C2\}$ ) wird durch die ISR des jeweiligen Treibers aufgerufen und übergibt die empfangenen Daten als Übergabeparameter an diese Funktion. Im ersten Schritt dieser Routine wird überprüft, ob Schreib- und Lesezeiger auf dieselbe Speicherzelle des FIFO-Speichers zeigen. Tritt dieser Fall ein, so existiert keine freie Speicherzelle zur Aufnahme der Daten im FIFO-Speicher. Die Signalisierung des Fehlers geschieht durch Setzen des **Error Flag**. Im fehlerfreien Fall oder im Fall des Speicherüberlaufs mit der FIFO-Konfiguration non-blocking mode, werden die Daten nun an der Position des Schreibzeigers in den FIFO-Speicher übertragen. Der Schreibzeiger wird anschließend um eine Position inkrementiert und das **RTS Flag** wird gesetzt.

Die Funktion `Data_Proc_Control_Loop()` wird periodisch durch die Main-Control-Loop aufgerufen. Anhand des **RTS Flag** wird jeder FIFO-Speicher überprüft, ob Daten darin abgelegt sind. Ist das **RTS Flag** gesetzt, so wird nun die vom uIP-Stack bereitgestellte Datenstruktur `UDP_Send_Buffer` mit den im FIFO-Speicher abgelegten Daten byteweise gefüllt. Die aktuelle Schreibposition in dieser Datenstruktur bestimmt die Variable `UDP_Data_Length`. Die Anzahl der zu übertragenden Speicherzellen errechnet sich aus der Distanz zwischen Lese- und Schreibzeiger eines FIFO-Speichers. Nach jedem Ablegen einer Speicherzelle in das UDP-Paket wird der Lesezeiger um **EINS** inkrementiert und die Distanz beider FIFO-Zeiger verringert sich dabei um **EINS**. Das Befüllen des UDP-Datenpaketes wird nun solange durchgeführt, bis die Distanz beider Zeiger **EINS** beträgt oder das Fassungsvermögen des UDP-Paketes erschöpft ist. Abschließend wird das **RTS Flag** zurückgesetzt, sofern keine versandbereiten Daten im FIFO-Speicher mehr vorhanden sind. Ist das **Error Flag** gesetzt, so wird versucht, den gesamten Inhalt des FIFO-Speichers in das UDP-Paket zu übertragen. Das Zurücksetzen des **Error Flag** geschieht bereits bei Setzen der Fehlersignalisierung in das UDP-Paket.

Eine Ausnahme bilden die FIFO-Speicher der UART basierenden Schnittstellen. Ein

Versand der Daten erfolgt erst, wenn die Distanz beider FIFO-Zeiger größer/gleich des konfigurierten `FIFO Character Charge` Parameters ist.

Um das UDP-Paket wie in Kapitel 3.2.5 beschrieben zu vervollständigen, werden abschließend noch an der Position 0 und 1 die Anzahl der nun folgenden Bytes auf Grundlage der Variablen `UDP_Data_Length` geschrieben. Der Wert des `Package Counter` wird an den Positionen 2 und 3 abgespeichert. Der uIP-Stack wird letztlich über die Versandbereitschaft des Datenpaketes informiert.

### 5.3.3 Integration des uIP-Stack

Die Programmteile des uIP-Stack befinden sich im Ordner `uip-1.0` des *mbed*-Softwareprojektes. Der Zugriff auf die Hardware des *mbed*-Mikrocontrollermoduls erfolgt durch den EMAC-Treiber `lpc17xx_emac.c` und den Treiber für Zeitinformationen `lpc17xx_systick.c`. Diese zusätzlichen Treiber befinden sich im Ordner `lpc1700` des *mbed*-Softwareprojektes.

Durch Editieren der Quelldatei `uip-conf.h` im Ordner des uIP-Stack können über die Präprozessoranweisung `#define` weitere Konfigurationsmerkmale gesetzt werden. Dem symbolischen Namen `UIP_CONF_UDP` wird hier die Konstante 1 zugewiesen. Diese Änderung ist notwendig, um die Unterstützung des UDP durch den uIP-Stack zu aktivieren. Notwendig sind hier auch die zusätzlichen Einträge, die die Header-Dateien mit der Definition der Einsprungpunkte inkludieren. Mittels der `#include` Anweisung wird hier auf die Header-Dateien `webserver.h` und `Network.h` verwiesen. Aus Sicht des uIP-Stack bilden diese Quelldateien die beiden Applikationen. Weitere uIP-Stack spezifischen Konfigurationsmöglichkeiten bietet die Quelldatei `uiptopt.h`.

Der Zugriff auf den uIP-Stack erfolgt durch den Programmteil `Network.c`. Noch vor dem Starten der Main-Control-Loop werden durch Aufruf der Funktion `Network_Init()` die Ethernetschnittstelle und die darauf zugreifenden Programmteile initialisiert. Hier erfolgt im ersten Schritt die Initialisierung von zwei Zeitgebern und des Ethernetblocks der MCU auf der Ebene der Hardwareschicht. Innerhalb der Applikationsschicht werden nun der uIP-Stack und der Webserver initialisiert. Auf Grundlage der globalen Umgebungsvariablen des Konfigurationsmoduls wird abschließend die UDP-Verbindung angelegt.

Durch Aufruf der Funktion `change_Host_data()` lassen sich die Netzwerk spezifischen Einstellungen der Multifunktions-Adapterplatine zur Laufzeit des Systems ändern. Die in den globalen Umgebungsvariablen enthaltenen Daten der eigenen IP-Adresse, MAC-Adresse und Subnetzmaske werden durch diese Funktion übernommen. Die IP-Adresse des UDP-Empfängers und der Ziel-Port lassen sich im laufenden Betrieb der Multifunktions-Adapterplatine durch Aufruf der Funktion `change_UDP_conn()` ändern. Dies geschieht durch das Auflösen der aktuellen UDP-Verbindung und dem Aufbau einer neuen Verbindung mit den aktuellen Werten der globalen Umgebungsvariablen.

Der Zugriff auf die Funktionen des uIP-Stack erfolgt über die uIP *raw* API. Hierbei handelt es sich um eine rein Ereignis basierende Schnittstelle zu anderen Programnteilen. Zu diesen Ereignissen zählen beispielsweise der Empfang eines Datenpaketes, die erfolgreiche Ankunft eines Datenpaketes beim Empfänger oder die Herstellung einer neuen Verbindung [14]. Bei Eintreten eines Ereignisses wird die Applikation aufgerufen. Die Applikation ist eine Funktion der Applikationsschicht und wird dem uIP-Stack in Form der `#define` Deklaration `UIP_UDP_APPCALL` für auf UDP basierende Verbindungen bzw. `UIP_APPCALL` für TCP Verbindungen mitgeteilt. Diese Applikation wird ebenso periodisch aufgerufen, um das Vorhandensein versandbereiter Daten zu überprüfen.

Die Definition der Einsprungpunkte findet in den Header-Dateien `webserver.h` und `Network.h` statt. Für das Senden von UDP-Paketen wird die Funktion `Network_app()` als Applikation definiert. Diese Funktion überprüft das Vorhandensein eines versandbereiten Datenpaketes und weist den uIP-Stack an, die im Puffer befindlichen Daten zu versenden. Durch den Aufruf der Funktion `Package_ready()` wird dabei ein boolescher Ausdruck gesetzt, der die Versandbereitschaft signalisiert.

Durch den periodischen Aufruf der Funktion `Network_Control_Loop()` durch die Main-Control-Loop wird auf Ereignisse anderer Protokolle reagiert. So werden Anfragen eines PING-Befehls des ICMP von dieser Funktion an den uIP-Stack weitergeleitet. Weitergeleitet werden ebenso die Anfragen zur Adressauflösung über das Address Resolution Protocol (ARP).

## 6 Implementierung des Webserver und Konfigurationsmöglichkeiten

### 6.1 Die Webserver Applikation

Alle Programmteile des Webserver, wie auch die Webseiten, befinden sich im Ordner `webserver` des *mbed*-Softwareprojektes.

Die Initialisierung der Webserver Applikation erfolgt nach der Initialisierung der Ethernetchnittstelle und noch vor dem Eintritt der Programmblaufkontrolle durch die Main-Control-Loop. Hierzu wird aus der `Network.c` die Funktion `httpd_init()`, die im Programmteil `httpd.c` deklariert ist, aufgerufen. Damit wird über die Funktionen des uIP-Stack ein Endpunkt mit der IP-Adresse des Stack auf Port 80 geschaffen. Als reine Server-Applikation implementiert, wartet der Webserver auf Anfragen von Clients und bedient diese daraufhin.

Der Empfang von TCP Paketen löst innerhalb des uIP-Stack ein Ereignis aus und veranlasst die Ausführung der deklarierten Applikation. Die Einsprungstelle dafür ist als Funktion in der Quelldatei `webserver.h` mit der Definition `UIP_APPCALL httpd_appcall` deklariert. Diese Funktion, die in der Quelldatei `httpd.c` implementiert ist, behandelt alle von einem Client ausgehenden Anfragen.

### 6.2 Aufbau der dynamischen Webseiten

Eine Aufgabe der Webserver Applikation ist es, die in HyperText Markup Language (HTML) vorliegende Dokumente über eine TCP Verbindung zu übertragen, damit sie innerhalb eines Webbrowser angezeigt werden können. Die Erweiterung des Webserver zur Webbrowser basierenden GUI erfordert darüber hinaus die Abarbeitung von dynamischen Webseiten. Erst die Interaktion mit dem Benutzer bietet die Möglichkeit zur Konfiguration der Multifunktions-Adapterplatine. Die statischen Webseiten mit der Dateiendung `html`, als auch die dynamischen Seiten mit der Endung `shtml`, befinden sich im Unterordner `httpd-fs` der Webserver Applikation.

Bei der Darstellung der dynamischen Webseiten wird durch diese Applikation auf das Prinzip der Server Side Includes (SSI) zurückgegriffen. Die dynamischen Webseiten sind

normale in HTML verfasste Dokumente, die über spezielle Befehle einen Sprung in Funktionen des Webservers veranlassen. Damit ist es möglich, statische Inhalte durch dynamische Ausgaben innerhalb des Dokumentes zu ersetzen und diese dem Client zukommen zu lassen.

Zu diesem Zweck sind die dynamischen Webseiten mit Einsprungmarken versehen, die bei Aufrufen der Webseite durch den Webbrowser einen Sprung in entsprechende Funktionen veranlassen. Diese Funktionen sind in der als *Web Server Script Interface* bezeichneten Quelldatei `httpd-cgi.c` implementiert. Eine Einsprungmarke innerhalb des Dokumentes der dynamischen Webseite beginnt mit der Zeichenkombination `%!`. Getrennt durch ein Leerzeichen, folgt der Funktionsaufruf, dem ein Parameter übergeben werden kann.

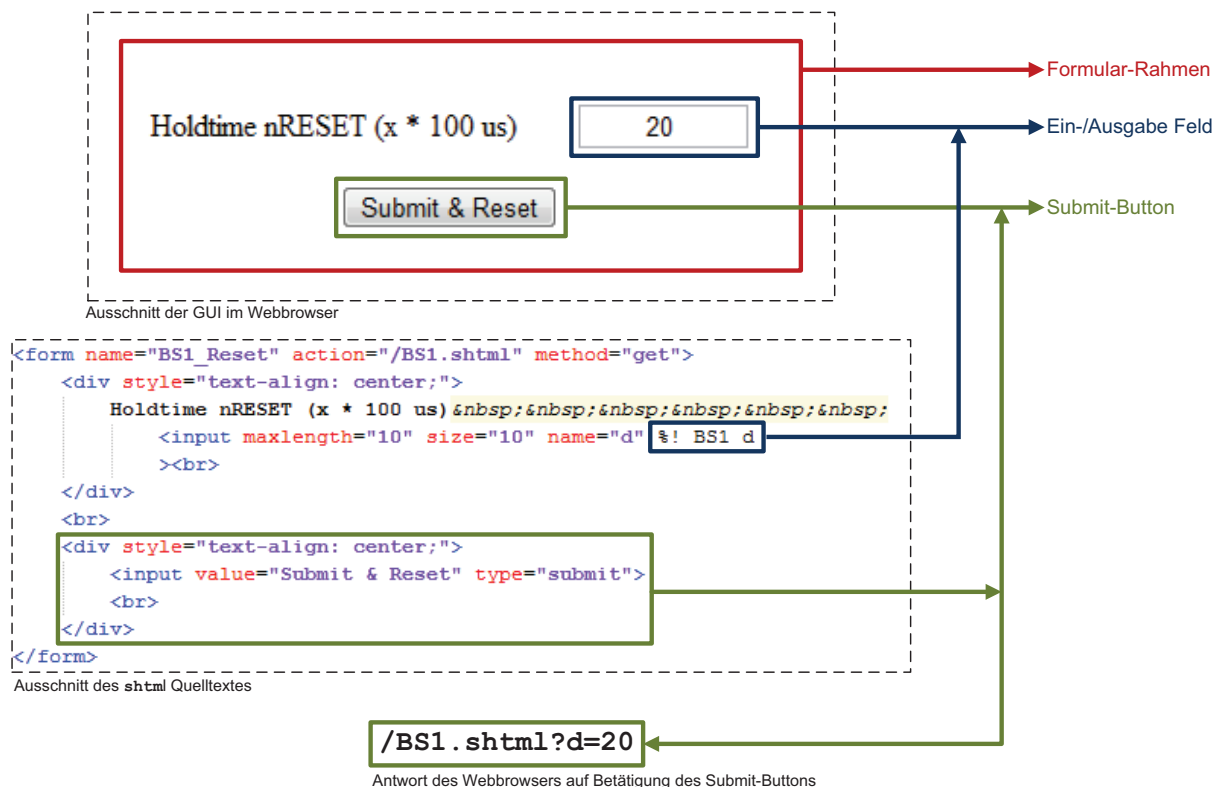


Abbildung 6.1: Ein- und Ausgabe durch eine dynamische Webseite

Durch die Funktion `HTTPD_CGI_CALL()` innerhalb der Quelldatei `httpd-cgi.c` wird die Zeichenkette des Funktionsaufrufs mit einer implementierten Funktion verknüpft. Anhand der als Zeichenkette übergebenen Parameter kann diese Funktion beispielsweise die Ausgabe eines aktuellen Wertes veranlassen.

In der vorliegenden Implementierung dieses Programmteiles existiert für jede dynamische Webseite eine eigene Einsprungfunktion. Diese Funktion verzweigt sich je nach Anfrage in zwei weitere Funktionen. Die erste Funktion mit den Präfix `generate` dient allein der Ausgabe von aktuellen Konfigurationsparametern. Die zweite Funktion mit dem Präfix `Config` stellt die schreibende Kommunikation mit dem Konfigurationsmodul her.

Für die Realisierung der Interaktivität der dynamischen Webseiten werden drei HTML Komponenten benutzt: Texteingabefelder, Radiobuttons und Checkboxes. Bei Aufruf der dynamischen Webseite wird in die zur Ausgabe vorgesehene Funktion verzweigt und anhand des Parameters die entsprechende Ausgabe nach Stand der globalen Umgebungsvariablen getätigt.

Zur Übernahme von eingegebenen Daten dient der Submit-Button. Dazu sind Gruppen von Eingabefelder in HTML als Formulare deklariert. Nach Betätigung des Submit-Buttons wird die Webseite erneut aufgerufen und die übernommenen Eingaben erscheinen im Webbrowser. Die getätigten Eingaben erscheinen ebenso, angeführt durch das Zeichen `?`, als nachfolgende Zeichenkette in der Adresszeile des Webbrowsers. Die Webserver Applikation erhält diese Daten anhand der Datenstruktur `httpd_state`. Diese Zeichenkette wird durch die `Config` Funktion ausgewertet und in entsprechende Konfigurationsbefehle gewandelt. Abbildung 6.1 zeigt am Beispiel der Konfiguration des Reset-Signals die Wechselwirkung zwischen Ausgabe und Eingabe von Daten über den Webserver.

Die Webseiten selbst werden dabei nicht als Textdokumente im Speicher der MCU abgelegt. Im Ordner `webserver` des `mbed`-Softwareprojektes befinden sich die beiden PERL-Scripte `makefsdata.pl` und `makestrings.pl`. Vor dem Compilieren des Softwareprojektes ausgeführt, erzeugen die Scripte die Quelldateien `httpd-fsdata.c` und `http-strings.c`. Diese Quelldateien enthalten in binärer Form die Webseiten als Konstanten in Form von Arrays.

## 6.3 Konfigurationsmöglichkeiten durch den Webserver

### 6.3.1 Grundlegende Funktionen

Mit Aufruf des Webservers über die IPv4 Adresse der initialisierten Ethernetschnittstelle der Multifunktions-Adapterplatine wird die statische Startseite im Webbrowser angezeigt. Die Menüleiste im Kopfbereich dieser Seite enthält die Links zu den dynamischen Seiten, die die Interaktion mit dem Nutzer ermöglichen. Abgesehen von den Menüpunkten **Home** und **General**, entspricht der Inhalt jeder dynamischen Webseite der Funktionalität der entsprechenden Hardwaregruppe. Der Gruppe `mbed` wird neben der ADC Funktionalität die Definition der Hardwaregruppen IDs zugeordnet.

Der Aufruf der dynamischen Webseite **General** zeigt in der oberen Hälfte eine Statusübersicht, die Auskunft über den Initialisierungszustand der Hardwaregruppen gibt. Der untere Teil der Seite enthält den Dialog zum Laden/Speichern von Konfigurationsdaten durch das EEPROM-Modul. Wie die Abbildung 6.2 zeigt, existiert hier die Auswahl zum Zurücksetzen der Konfiguration auf die Standardwerte. Mit Betätigung des Submit-Buttons wird die gewählte Aktion unverzüglich ausgeführt. Ist Breakout-Socket 1 nicht im Betriebsmodus **FLASH EEPROM** initialisiert, bleibt die Aktion Laden/Speichern wirkungslos.

---

---

**Multi IF Configuration**

☒ Load Configuration from FLASH EEPROM  
☐ Save Configuration to FLASH EEPROM  
☐ Reset to Default Configuration

**Abbildung 6.2:** Webserver-Dialog zur Verwaltung der Gesamtkonfiguration

Die Menüpunkte BS1 bis BS4 rufen die dynamischen Webseiten zur Konfiguration der Breakout-Sockets 1 bis 4 auf. Auf diesen Seiten erscheint stets als erster Dialog die Auswahl des Betriebsmodus eines Breakout-Socket, wie es Abbildung 6.3 zu sehen ist. Hier erfolgt die Auswahl der entsprechenden Schnittstellenfunktionalität. Der Modus **Undefined** wird hingegen stets nach De-Initialisierung einer aktiven Schnittstelle eingenommen. Ist das entsprechende Breakout-Socket in irgendeinen Modus initialisiert, so wird durch Wahl des Modus **Undefined** der Socket unverzüglich de-initialisiert. Eine Initialisierung in diesem Betriebsmodus ist nicht möglich.

---

---

**Functional Type**

Undefined ☐  
RS-232 ☐  
RS-422 ☐  
RS-485 ☐  
I2C ☒

---

---

**Abbildung 6.3:** Webserver-Dialog zur Auswahl des Betriebsmodus

Durch einen Dialog lässt sich der FIFO-Speicher einer jeden Schnittstellenfunktionalität entweder im **blocking mode** oder im **non-blocking mode** betreiben. Eine Veränderung dieser Konfiguration wirkt sich unmittelbar auf die initialisierte Schnittstelle aus.

Das Initialisieren und De-Initialisieren der Breakout-Sockets, der CAN-Schnittstelle und des ADC erfolgen mit einem Dialog am Ende der entsprechenden Webseite. Die getätigte Konfiguration dient dann der Initialisierung der entsprechenden Funktionalität.

### 6.3.2 Konfiguration der auf UART basierenden Schnittstellen

Der Dialog, wie in Abbildung 6.4 gezeigt, dient der Konfiguration aller auf UART basierenden Schnittstellen. Einstellungen zur Baudrate, der Wortlänge eines Zeichens, die

Wahl der Parität und die Anzahl der Stopbits konfigurieren den in der MCU integrierten UART-Controller. Die Änderung dieser Werte wirkt sich erst bei Initialisierung bzw. Re-Initialisierung der entsprechenden Schnittstelle aus.

---

**UART**

|                                                                            |                                                                                           |
|----------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| Baudrate (bit per second)                                                  | <input type="text" value="200000"/>                                                       |
| Wordlength (5-8 bit)                                                       | <input type="text" value="8"/>                                                            |
| Parity bit                                                                 | <input type="radio"/> EVEN <input type="radio"/> ODD <input checked="" type="radio"/> OFF |
| Stopbits (1-2 bit)                                                         | <input checked="" type="radio"/> One <input type="radio"/> Two                            |
| FIFO Character Charge                                                      | <input type="text" value="1"/>                                                            |
| FIFO Release Time (x * 100 us)                                             | <input type="text" value="2000"/>                                                         |
| FIFO overflow behaviour                                                    | <input checked="" type="radio"/> blocking <input type="radio"/> non-blocking              |
| <input type="button" value="Submit"/> <input type="button" value="Reset"/> |                                                                                           |

---

**Abbildung 6.4:** Webserver-Dialog zur UART-Konfiguration

Der Parameter **FIFO Character Charge** definiert die Mindestmenge an Zeichen, die im FIFO-Speicher vorrätig sein müssen, um diese Datenbytes in das zu sendende UDP-Paket zu schreiben. Dieser Parameter kann Werte im zweistelligen Bereich annehmen. Die **FIFO Release Time** gibt den Zeitintervall in Schritten zu  $100\mu s$  an, in der die restlichen im FIFO-Speicher abgelegten Zeichen unterhalb der Mindestvorhaltemenge zum Versand freigegeben werden. Die Einstellung dieses Parameters ist dabei stark abhängig von der verwendeten Baudrate und daher mit Bedacht zu wählen, da eine zu klein gewählte Einstellung dieses Parameters die Wirkung des Parameters **FIFO Character Charge** wirkungslos macht. Änderungen an diesen beiden Parametern wirken sich unmittelbar auf die Funktion der initialisierten auf UART basierenden Schnittstelle aus.

### 6.3.3 Konfiguration der SPI-Schnittstelle

Die Konfiguration der generischen SPI-Schnittstellen im Master Mode erfolgt anhand der in der Abbildung 6.5 gezeigten Dialoges. Die Wahl des SCLK, des SPI Mode und der Abfragefrequenz werden stets bei Initialisierung und Re-Initialisierung wirksam. Änderungen am Wert des 32 Bit langen in hexadezimalen Zeichen eingegebenen Kommandowortes werden bei initialisierter SPI Schnittstelle sofort umgesetzt. Wird die SPI Schnittstelle mit einer Abfragefrequenz von  $0Hz$  initialisiert, so erfolgt auf Betätigung des Submit-Buttons eine Einzelabfrage.

---

**SPI**

|                                                                                                      |                                                                      |
|------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------|
| SPI Clock Frequency (in Hz)                                                                          | <input type="text" value="800000"/>                                  |
| Clock Phase                                                                                          | <input checked="" type="radio"/> CPHA=1 <input type="radio"/> CPHA=0 |
| Clock Polarity                                                                                       | <input checked="" type="radio"/> CPOL=1 <input type="radio"/> CPOL=0 |
| Request Frequency (in Hz)                                                                            | <input type="text" value="10000"/>                                   |
| Command Word (HEX)                                                                                   | <input type="text" value="10EAF660"/>                                |
| FIFO overflow behaviour <input checked="" type="radio"/> blocking <input type="radio"/> non-blocking |                                                                      |
| <input type="button" value="Submit"/> <input type="button" value="Reset"/>                           |                                                                      |
| Holdtime nRESET (x * 100 us) <input type="text" value="20"/>                                         |                                                                      |
| <input type="button" value="Submit &amp; Reset"/>                                                    |                                                                      |

---

**Abbildung 6.5:** Webserver-Dialog zur SPI-Konfiguration

Durch den separaten RESET-Dialog wird der LOW-Pegel des RESET-Signals für die gewählte Dauer in Schritten zu  $100\mu s$  manuell ausgelöst.

### 6.3.4 Konfiguration der $I^2C$ -Schnittstelle

Die beiden verfügbaren  $I^2C$ -Schnittstellen werden anhand des in der Abbildung 6.6 dargestellten Dialoges konfiguriert. Die als dezimale Werte angegebenden Adressen sind in einem Bereich von 0-127 gültig. Abgesehen von der Einstellung der Parameter **FIFO overflow behaviour** und **Commandbyte**, werden alle Änderungen erst bei Initialisierung bzw. Re-Initialisierung der Schnittstelle wirksam. Durch Betätigen des Submit-Buttons wird stets eine Einzelabfrage durch die als Master definierte Schnittstelle durchgeführt. Eine im Slave Mode initialisierte  $I^2C$ -Schnittstelle verhält sich dagegen rein passiv.

---

**I2C**

|                                                                                                      |                                                  |                                                         |
|------------------------------------------------------------------------------------------------------|--------------------------------------------------|---------------------------------------------------------|
| Mode                                                                                                 | <input type="radio"/> Slave                      | <input checked="" type="radio"/> Master                 |
| Baudrate                                                                                             | <input type="radio"/> 100 KBit/s (Standard Mode) | <input checked="" type="radio"/> 400 KBit/s (Fast Mode) |
| Slave Mode:                                                                                          | own address (DEC)                                | <input type="text" value="16"/>                         |
| Master Mode:                                                                                         | destination address (DEC)                        | <input type="text" value="32"/>                         |
|                                                                                                      | Commandbyte (HEX)                                | <input type="text" value="FF"/>                         |
| FIFO overflow behaviour <input checked="" type="radio"/> blocking <input type="radio"/> non-blocking |                                                  |                                                         |
| <input type="button" value="Submit"/> <input type="button" value="Reset"/>                           |                                                  |                                                         |

---

**Abbildung 6.6:** Webserver-Dialog zur  $I^2C$ -Konfiguration

### 6.3.5 Konfiguration der CAN-Schnittstelle

Die grundlegende Konfiguration der CAN-Schnittstelle erfolgt im Dialog der ersten Sektion der CAN-Webseite, wie es Abbildung 6.7 zeigt. Einzig die Übernahme der Änderung der Baudrate macht eine Initialisierung bzw. Re-Initialisierung der CAN-Schnittstelle notwendig. Die Aktivierung und De-Aktivierung des Akzeptanzfilters wirkt sich unmittelbar auf die Funktion der CAN-Schnittstelle aus.

---

**General**

Baudrate (bits per second)

100000

Acceptance Filter

☐ BYPASS

☒ Filter ON

FIFO overflow behaviour

☒ blocking

☐ non-blocking

Submit

**Abbildung 6.7:** Webserver-Dialog zur generellen Konfiguration der CAN-Schnittstelle

In den folgenden vier Sektionen der CAN-Webseite werden über die Dialoge die vier Listen des Akzeptanzfilters beschrieben. Getätigte Filtereinträge bleiben gespeichert und können über die Checkbox aktiviert bzw. de-aktiviert werden. Die Eingabe der Message IDs kann in beliebiger Reihenfolge erfolgen. Das Betätigen des Submit-Buttons führt zur unmittelbaren Änderung des Akzeptanzfilters bei aktiver CAN-Schnittstelle. Das Sortieren der einzelnen Filtereinträge übernimmt der CAN-Treiber.

---

**Group ID Filter for Extended Frame**

Filter active

☐

Start Message ID

15000

End Message ID

15010

☐

14000

14010

Submit

Reset

Filter active

☒

Start Message ID

13000

End Message ID

13010

☒

12000

12010

Submit

Reset

**Abbildung 6.8:** Webserver-Dialog zur Konfiguration des CAN Gruppen-ID Akzeptanzfilters

Bei der Definition der Gruppen-IDs für den Akzeptanzfilter ist allerdings stets darauf zu achten, dass der Wert der End Message ID größer der Start Message ID ist. Abbildung 6.8 zeigt exemplarisch die Einstellungen des Gruppen ID Filters anhand des web-basierenden Dialoges.

### 6.3.6 Konfiguration der Netzwerkeinstellungen

Die Konfiguration der Ethernetschnittstelle der Multifunktions-Adapterplatine unterteilt sich in zwei Bereiche. Der erste mit „Server address“ betitelte Bereich erlaubt die Konfiguration der Netzwerkeinstellungen der Multifunktions-Adapterplatine, so wie es die Abbildung 6.9 zeigt. Die Übernahme der in diesem Dialog getätigten Änderungen wirken sich nicht zwangsläufig unmittelbar aus. So lassen sich Netzwerkeinstellungen für einen bevorstehenden Einsatz in einem anderen Netzwerk definieren und speichern. Das Aktivieren der Checkbox „Apply immediately“ übernimmt dagegen alle getätigten Änderungen sofort durch eine Re-Initialisierung des uIP-Stack. Eine Änderung der IPv4-Adresse führt dann zwangsläufig zum Verlust der Verbindung zur Webserver-Applikation. Die Angabe des Quell UDP-Port ist rein optional. Bei Parametrisierung mit dem Wert 0 erfolgt das Binden des Quell Port durch den uIP-Stack selbständig.

---

**Server address**

|                 |       |   |     |   |     |   |     |   |    |   |    |
|-----------------|-------|---|-----|---|-----|---|-----|---|----|---|----|
| MAC address     | 0A    | : | 16  | : | 2C  | : | 58  | : | 78 | : | C8 |
| IPv4            | 139   | . | 30  | . | 204 | . | 223 |   |    |   |    |
| Subnet Mask     | 255   | . | 255 | . | 255 | . | 0   |   |    |   |    |
| UDP Port (opt.) | 45678 |   |     |   |     |   |     |   |    |   |    |

☐ Apply immediately

Submit

Reset

---

**Abbildung 6.9:** Webserver-Dialog zur Server-Netzwerkeinstellung

Alle im Dialog „Client address“ bestätigten Eingaben werden unmittelbar übernommen. Die hier getätigten Eingaben definieren den Ziel-Teilnehmer innerhalb des Netzwerkes, so wie es die Abbildung 6.10 zeigt. Der in der Multifunktions-Adapterplatine eingehende Datenstrom wird an das Ziel, welches sich durch die IPv4-Adresse und Port-Nummer identifiziert, in Form von UDP-Paketen weitergeleitet. Das Aktivieren der Checkbox „Broadcast“ versendet die so entstandenen UDP-Datenpakete an alle Teilnehmer des Netzes.

---

**Client address**

|           |                          |   |    |   |     |   |     |
|-----------|--------------------------|---|----|---|-----|---|-----|
| Broadcast | <input type="checkbox"/> |   |    |   |     |   |     |
| IPv4      | 139                      | . | 30 | . | 204 | . | 217 |
| UDP Port  | 8000                     |   |    |   |     |   |     |

Submit

Reset

---

**Abbildung 6.10:** Webserver-Dialog zur Client-Netzwerkeinstellung

### 6.3.7 Konfiguration des ADC

Der über den Link `mbed` aufgerufene Dialog erlaubt die Kontrolle des internen ADC der MCU. Alle getätigten Eingaben zur Wahl der ADC-Kanäle und der Abtastfrequenz werden bei Initialisierung bzw. Re-Initialisierung des ADC übernommen. Die Wahl der Abtastfrequenz ist auf  $f_t = 200\text{kHz}$  bei Auswahl nur eines ADC-Kanals limitiert. Abbildung 6.11 zeigt den Dialog des Webservers zur Konfiguration des ADC.

---

**Internal Analog to Digital Converter**

|                       |                                     |
|-----------------------|-------------------------------------|
| ADC Channel 1         | <input checked="" type="checkbox"/> |
| ADC Channel 2         | <input type="checkbox"/>            |
| ADC Channel 3         | <input checked="" type="checkbox"/> |
| ADC Channel 4         | <input type="checkbox"/>            |
| Sample Frequency (Hz) | <input type="text" value="10000"/>  |

FIFO overflow behaviour    ☒ blocking    ☐ non-blocking

ADC    ☐ ON    ☒ OFF

Abbildung 6.11: Webserver-Dialog zur Ansteuerung des ADC

### 6.3.8 Konfiguration der GPIO-Funktionalität

Breakout-Socket 1 stellt die einfache Funktionalität zum Ansteuern von fünf digitalen Ausgängen zur Verfügung. Wie es Abbildung 6.12 zeigt, erfolgt die Deklaration der fünf Datenleitungen durch Aktivieren/De-Aktivieren der entsprechenden Checkboxes im Betriebsmodus GPIO in den Zuständen HIGH oder LOW. Ist das Breakout-Socket 1 in diesem Betriebsmodus initialisiert, erfolgt die Übernahme der Änderungen mit sofortiger Wirkung.

---

**GPIO**

|          |                                     |
|----------|-------------------------------------|
| BB1 PIN3 | <input type="checkbox"/>            |
| BB1 PIN4 | <input checked="" type="checkbox"/> |
| BB1 PIN5 | <input checked="" type="checkbox"/> |
| BB1 PIN6 | <input checked="" type="checkbox"/> |
| BB1 PIN7 | <input type="checkbox"/>            |

---

Abbildung 6.12: Webserver-Dialog zur Steuerung der GPIO-Funktionalität

## 7 Test der Hardwareplattform und Validierung der Software

### 7.1 Allgemeine Funktionsprüfung

Die in diesem Kapitel beschriebenen Tests dienen der Funktionsüberprüfung der Multifunktions-Adapterplatine. Das Zusammenspiel zwischen Hardware des Base-Boards und der realisierten Breakout-Boards mit der *mbed*-Software auf Ebene der Hardware-schicht und Applikationsschicht soll so hinreichend nachgewiesen werden.

Als unverzichtbare Werkzeuge zur Funktionsprüfung erwiesen sich die innerhalb des Service-Moduls implementierten Testfunktionen und die Software *Wireshark*. Das Aktivieren der Testfunktion versetzt eine oder mehrere der drei verfügbaren auf UART basierenden Schnittstellen in den Sendemodus. Dadurch wird die Überprüfung der kalkulierten Baudrate möglich, wie der nachstehende Test zeigt.

*Wireshark* analysiert die im Netzwerk befindlichen Datenpakete. Durch Aktivieren von Filtermöglichkeiten lassen sich hier die ausschließlich von der Multifunktions-Adapterplatine gesendeten UDP-Datenpakete anzeigen. Anhand der Aufzeichnung dieses Datenverkehrs im LAN lassen sich die einzelnen Datenpakete hinsichtlich des Inhaltes und der Korrektheit des UDP-Nutzdatenprotokolls hin überprüfen. Abbildung 7.1 zeigt die Analyse eines dieser Datenpakete (hier: ohne Data Length Counter).

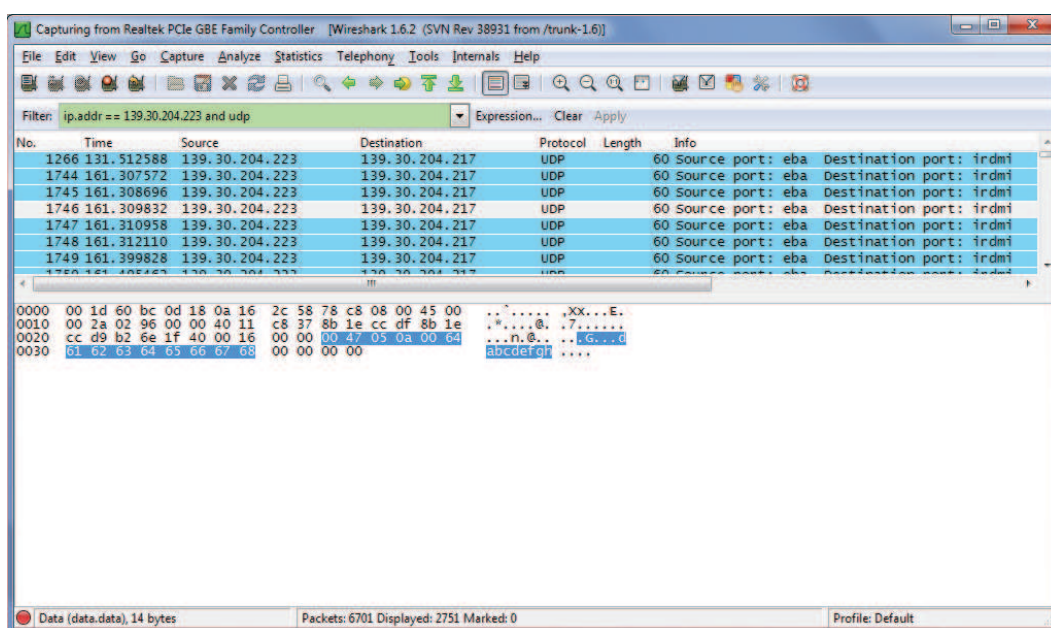


Abbildung 7.1: Software *Wireshark* zur Analyse der versendeten UDP-Pakete

## 7.2 Validierung der Baudrate der auf UART basierenden Schnittstellen

Die Berechnung der Registerwerte des Baudratengenerators der MCU auf Grundlage der konfigurierten Übertragungsgeschwindigkeit kann durch den Treiber auf zwei unterschiedliche Arten durchgeführt werden. Die erste Methode führt die Kalkulation der Registerwerte ausschließlich mit den Mitteln der Ganzzahlarithmetik durch.

Die zweite Variante zur Kalkulation der Registerwerte basiert auf der Benutzung des in dem UART-Controller integrierten **fractional divider**. Die Berechnung der entsprechenden Werte nach dieser Methode garantiert eine Abweichung der erzeugten Baudrate von maximal 1,1% vom konfigurierten Sollwert [19].

In diesem Test werden beide Methoden zur Ansteuerung des Baudratengenerators implementiert und der Ausgabe der tatsächlichen Baudrate gegenübergestellt. Dazu wird einer der UART-tauglichen Breakout-Sockets ohne installiertes Breakout-Board in den Betriebsmodus RS-232, RS-422 oder RS-485 initialisiert. Durch Starten eines Ausgabetests zum permanenten Senden des Zeichens U wird bei Nutzung der 8-N-1 Codierung ein Rechtecksignal erzeugt. Die Frequenz des TXD-Signals wird am Pin des Breakout-Sockets mit dem Oszilloskop *LeCroy 9350AM* gemessen. Die Übereinstimmung mit dem konfigurierten Sollwert der Baudrate ist gegeben, wenn die Frequenz des gemessenen Signals genau der Hälfte des Sollwertes entspricht. Die Tabelle 7.1 enthält die innerhalb einer Messreihe aufgenommenen Messwerte für verschiedenen Baudraten.

| Baudrate (Bit/s) | ohne <b>fractional divider</b> |             | mit <b>fractional divider</b> |             |
|------------------|--------------------------------|-------------|-------------------------------|-------------|
|                  | Messwert in Hz                 | Fehler in % | Messwert in Hz                | Fehler in % |
| 9'600            | 4'800                          | 0           | 4'800                         | 0           |
| 19'200           | 9'615                          | 0,16        | 9'645                         | 0,47        |
| 57'600           | 28'935                         | 0,47        | 28'935                        | 0,47        |
| 115'200          | 57'870                         | 0,47        | 57'870                        | 0,47        |
| 230'400          | 115'740                        | 0,47        | 115'741                       | 0,47        |
| 300'000          | 156'250                        | 4,17        | 150'240                       | 0,16        |
| 400'000          | 208'333                        | 4,17        | 200'895                       | 0,45        |
| 460'800          | 240'385                        | 4,33        | 230'825                       | 0,18        |
| 600'000          | 312'500                        | 4,17        | 300'480                       | 0,16        |
| 700'000          | 390'625                        | 11,61       | 350'010                       | 0           |
| 800'000          | 446'228                        | 11,56       | 400'640                       | 0,16        |
| 921'600          | 520'833                        | 13,03       | 461'600                       | 0,17        |

**Tabelle 7.1:** Messwerte zur Überprüfung der tatsächlich erzeugten Baudrate

Der Test zeigt, dass beide Varianten zur Berechnung der Registerwerte zur Generierung der Baudrate für die RS-232 Schnittstelle geeignet sind. Erst bei Nutzung einer Datenübertragungsrate ab  $300kBit/s$  zeigt die Verwendung des **fractional divider**

einen deutlichen Vorteil bei der Erzeugung einer akkuraten Baudrate. Der implementierte UART-Treiber verwendet ausschließlich die zweite Methode zur Berechnung der erforderlichen Werte für den Baudratengenerator.

### 7.3 Test der Gesamtfunktionalität

Dieser Test dient der Funktionsprüfung der drei realisierten Breakout-Boards der RS-232 Schnittstelle und der Validierung der *mbed*-Software.

Das Base-Board wird mit den RS-232 Breakout-Boards bestückt. Zwei Breakout-Boards werden dabei über entsprechende Kabel mit der RS-232 Schnittstelle der Test-PCs verbunden. Die auf den PCs ausgeführte Software *HTerm* erlaubt das periodische Senden einer definierten Zeichenkette im Intervall von  $100ms$ . Die Zeichenkette besteht aus den 26 kleinen Buchstaben des deutschen Alphabets. Abbildung 7.2 zeigt den Dialog zum periodischen Senden der Zeichenkette.

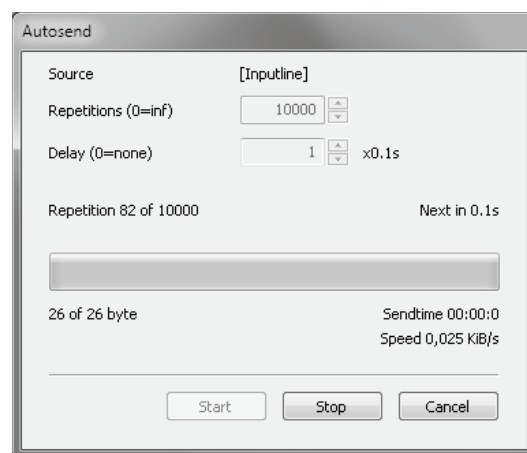
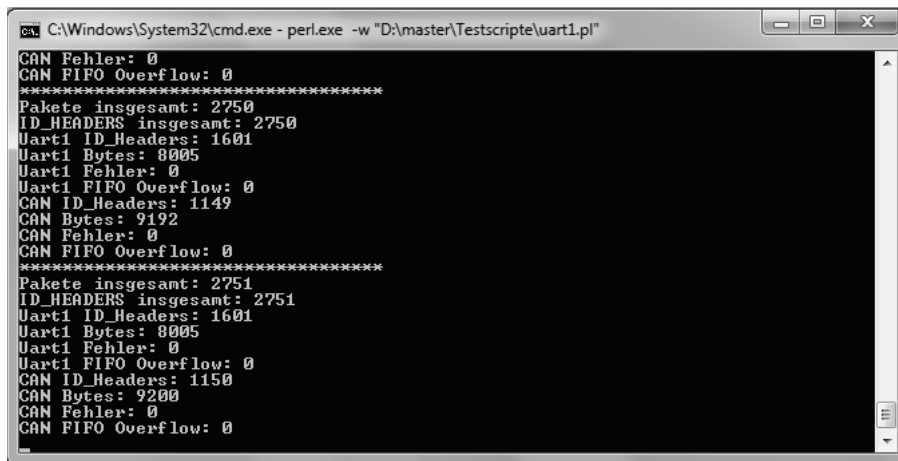


Abbildung 7.2: Software *HTerm* als RS-232 Testsender

Das Ziel, das die durch die Multifunktions-Adapterplatine versendeten UDP-Datenpakete empfängt, wird in Form eines Scripts in der Interpretersprache PERL realisiert. Dieses Script übernimmt die Rolle des Client und empfängt die UDP-Pakete und wertet den Inhalt der empfangenen Daten aus. Die ersten 4 Bytes des **Data Length** und **Package Counter** werden dabei nicht berücksichtigt. Entsprechend der ID wird der Inhalt der empfangenen Datennutzlast mit dem Sollwert verglichen. Ist ein Paket vollständig durch das Script analysiert worden, erfolgt auf der Konsole eine Ausgabe zur fortlaufenden Teststatistik. Abbildung 7.3 zeigt die Ausgabe durch das Testscript.



```
C:\Windows\System32\cmd.exe - perl.exe -w "D:\master\Testscript\uart1.pl"
CAN Fehler: 0
CAN FIFO Overflow: 0
*****
Pakete insgesamt: 2750
ID_HEADERS insgesamt: 2750
Uart1 ID-Headers: 1601
Uart1 Bytes: 8005
Uart1 Fehler: 0
Uart1 FIFO Overflow: 0
CAN ID-Headers: 1149
CAN Bytes: 9192
CAN Fehler: 0
CAN FIFO Overflow: 0
*****
Pakete insgesamt: 2751
ID_HEADERS insgesamt: 2751
Uart1 ID-Headers: 1601
Uart1 Bytes: 8005
Uart1 Fehler: 0
Uart1 FIFO Overflow: 0
CAN ID-Headers: 1150
CAN Bytes: 9200
CAN Fehler: 0
CAN FIFO Overflow: 0
```

Abbildung 7.3: PERL-Script als Testempfänger

Zu jedem Testdurchlauf werden die RS-232 Schnittstellen der Multifunktions-Adapterplatine und die der Test-PCs entsprechend der Baudrate und Codierung konfiguriert. Die Übernahme der geänderten Einstellungen der Breakout-Sockets wird stets durch De-Initialisierung/Initialisierung durch die GUI durchgeführt. Ein RESET des *mbed*-Mikrocontrollermoduls wird nicht ausgelöst. Beide Test-PCs senden möglichst zeitgleich tausend mal die entsprechende Zeichenkette. Die Ausgabe der Teststatistik nach dem Empfang des letzten Datenpaketes durch das Script kann dann zur Auswertung herangezogen werden.

Um die korrekte Funktion der Breakout-Boards im Sendebetrieb zu überprüfen, wird innerhalb jedes Testdurchlaufes mit der aktuellen Konfiguration mehrmals eine Zeichenkette mit 64 Elementen durch das Servicemodul gesendet. Die Software *HTerm* empfängt diese Zeichenkette und eine Sichtprüfung gibt Aufschluss über den korrekten Versand der Daten.

Mit folgenden Baudraten wurden Testdurchläufe durchgeführt:

- 1'200*Bit/s*
- 4'800*Bit/s*
- 9'600*Bit/s*
- 19'200*Bit/s*
- 56'000*Bit/s*
- 115'200*Bit/s*

Die Konfiguration der Codierung wurde zwischen jeden einzelnen Testdurchlauf zwischen 8-N-1 (8 Bit Wortlänge, keine Parität, 1 Stopbit) und 7-E-2 (7 Bit Wortlänge, gerade Parität, 2 Stopbits) gewechselt.

Ab einer Datenübertragungsrate von  $56'000 \text{ Bit/s}$  wurden Übertragungsfehler bei beiden RS-232 Breakout-Boards durch die Teststatistik des Scripts gemeldet. Die Analyse dieser Fehlerfälle durch Anhalten des Scripts ergab jedoch, dass die Datennutzlast korrekt übertragen wurde. Stattdessen traten in diesen Testdurchläufen erstmals der Tausch der Empfangsreihenfolge der UDP-Datenpakete beim Empfänger auf. Von den insgesamt ca. 52'000 Paketen kam es in diesem Fall zu durchschnittlich 4 Vertauschungen. Mit Verdoppeln der Baudrate verdreifacht sich dieser Fehler bei konstanter Paketanzahl.

Durch Verwendung des Konfigurationsparameters **FIFO Character Charge** konnte dieser Fehler allerdings reduziert bzw. eliminiert werden. Bei einer Baudrate von  $115'200 \text{ Bit/s}$  und der Konfiguration **FIFO Character Charge** = 2, wurden von den 10'397 Paketen nur 3 in der Empfangsreihenfolge vertauscht. Mit der Konfiguration **FIFO Character Charge** = 5 kam es zu keinen Vertauschungen mehr.

Abgesehen von dem o.g. Verhalten ergab die Auswertung der einzelnen Teststatistiken, dass unter jeder Konfiguration alle Zeichen korrekt durch das Script empfangen wurden. Die Funktionsfähigkeit der Komponenten Breakout-Board und *mbed*-Software der Multifunktions-Adapterplatine ist damit gewährleistet. Der Empfang der 64 Elemente umfassenden Zeichenkette durch die Test-PCs verlief ebenso erfolgreich. Ebenso wird das Datennutzlastprotokoll durch das FIFO-Softwaremodul korrekt aufgebaut. Zu einem Überlauf des FIFO-Speichers kam es in keinen der Testdurchläufe.

## 7.4 Test der CAN-Schnittstelle

Die Funktionsprüfung der CAN-Schnittstelle der Multifunktions-Adapterplatine wurde mit Hilfe des Gerätes *PCAN-USB* der Firma PEAK-System Technik GmbH durchgeführt. Mit diesem Gerät erfolgt der Zugriff auf den CAN-Bus durch die USB-Schnittstelle des PC. Die ebenfalls von dieser Firma stammende Software *PCAN-View* ermöglicht das Versenden von CAN-Nachrichten unterschiedlicher Länge bei verschiedenen Übertragungsgeschwindigkeiten. Der Versand der Nachrichten kann dabei im Standard-Frame-Format, als auch im Extended-Frame-Format in einem definierten Zeitintervall erfolgen. Abbildung 7.4 zeigt den Dialog zum Versenden einer CAN-Nachricht durch die PC-Software.

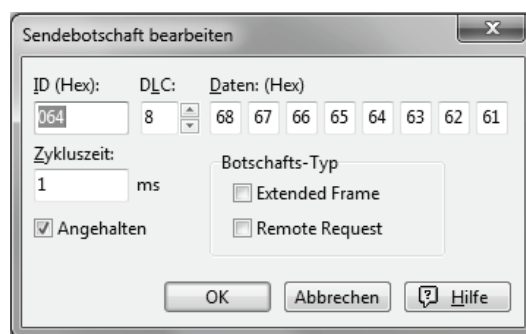


Abbildung 7.4: Dialog der Software *PCAN-View*

Für die Durchführung der Funktionsprüfung werden CAN-Nachrichten mit einer Datennutzlast von 8 aufeinanderfolgenden Zeichen verwendet. Das Senden der Nachrichten erfolgt im kleinstmöglich wählbaren Intervall von  $1ms$  jeweils im Standard- und Extended-Frame-Format.

Ausgewertet werden die durch die Multifunktions-Adapterplatine versendeten UDP-Pakete durch das bereits im Test zuvor benutzte PERL-Script. Auch in diesem Test wird die Datennutzlast der CAN-Nachricht mit dem Sollwert verglichen und die Teststatistik als Ergebnis auf der Konsole ausgegeben.

In jedem Testdurchlauf wurden über das *PCAN-USB* ca. 54'000 CAN-Nachrichten an die initialisierte CAN-Schnittstelle der Multifunktions-Adapterplatine versandt. Die Anzahl der durch das Script empfangenen Pakete wurde mit der Anzahl der durch die Software *Wireshark* registrierten Pakete verglichen, wie auch im Test zuvor. Der Gesamttest verlief unter Benutzung unterschiedlicher Übertragungsgeschwindigkeiten, die durch De-Initialisierung/Initialisierung der CAN-Schnittstelle durch die GUI der *mbed*-Software konfiguriert wurde.

Die Testdurchläufe wurden bei folgenden Übertragungsgeschwindigkeiten durchgeführt:

- $100kBit/s$
- $125kBit/s$
- $250kBit/s$
- $500kBit/s$
- $1MBit/s$

Die Funktionsüberprüfung der CAN-Schnittstelle der Multifunktions-Adapterplatine fällt in allen Punkten positiv aus. Es konnten keine Fehler bei den verwendeten Konfigurationen festgestellt werden.

## 7.5 Test mit Hilfe des Schiffsführungssimulators

Der Test mit Hilfe des Schiffsführungssimulators stellt einen Langzeittest zur Überprüfung der Funktionalität der drei RS-232 Breakout-Boards und der *mbed*-Software dar. Der Schiffsführungssimulator versendet über bis zu drei RS-232 Schnittstellen mit Baudraten von  $38'400Bit/s$  und  $4'800Bit/s$  nautische Datensätze in dem von der National Marine Electronics Association (NMEA) spezifizierten Kommunikationsstandard NMEA 0183.

Dieser eingehende Datenstrom wird in das UDP-Datenprotokoll der Multifunktions-Adapterplatine umgesetzt und an den Empfänger, einem Industrie-PC mit *xPC-Target*-Applikation, versandt. Der Konfigurationsparameter **FIFO Character Charge** wurde dabei in einem Bereich von 1 bis 30 Zeichen variiert. Der Konfigurationsparameter **Release Time** wurde im Rahmen dieses Tests in einem Bereich von  $50ms$  bis  $200ms$  verändert.

Die *xPC-Target*-Applikation wertet die über das LAN empfangen Datensätze aus. Unvollständige oder fehlerhafte Datensätze werden hier durch entsprechende Fehlermeldungen signalisiert. Aber auch in diesem Test zeigt sich die Hardware und Software der Multifunktions-Adapterplatine als zuverlässiger Schnittstellenwandler. Die Problematik des Vertauschens der Empfangsreihenfolge einzelner UDP-Datenpakete trat im Laufe dieses Tests nicht auf. Die übrigen Testkriterien wurden vollständig erfüllt.

## 8 Zusammenfassung und Ausblick

### 8.1 Zusammenfassung

Das Ziel dieser Masterarbeit bestand in der Konzeption und Entwicklung eines universellen Schnittstellenwandlers. Diese Multifunktions-Adapterplatine soll dem Anschluss verschiedener Kommunikationsteilnehmer mit unterschiedlichen physischen Schnittstellen dienen.

Im zweiten Kapitel wurden die Grundlagen der zu realisierenden Schnittstellentypen dargelegt. Das dort vermittelte Grundverständnis der logischen und physikalischen Eigenschaften dieser Schnittstellen ist die Voraussetzung zur Entwicklung der Multifunktions-Adapterplatine.

Das hardware- und softwaretechnische Konzept der Multifunktions-Adapterplatine wurde in Kapitel 3 ausführlich vorgestellt. Der grundlegende Aufbau und die Funktion des Base-Boards und der Breakout-Boards wurden dort beschrieben. Die Einbindung der Stromversorgung über das PoE-Verfahren und Maßnahmen zur Sicherstellung der EMV in das Hardwarekonzept wurden dort erörtert. In diesem Kapitel wurde ebenso das *mbed*-Mikrocontrollermodul als zentrales Element vorgestellt. Die Beschreibung des Softwarekonzepts und die modellhafte Abbildung der Software in Hardware- und Applikationsschicht war Inhalt des zweiten Teils dieses Kapitels.

Das vierte Kapitel beschrieb die Realisierung der Hardware der Multifunktions-Adapterplatine. An dieser Stelle wurden Aufbau und Funktion einzelner elektronischer Schaltungen und Bauelemente beschrieben.

Das fünfte Kapitel widmete sich der Implementierung des *mbed*-Softwareprojektes. Zunächst wurde dort der Aufbau und Abhängigkeit der *mbed*-Software beschrieben. Danach wurden dort Details zur Implementierung der Treiber der Hardwareschicht und die der Module der Applikationsschicht dargelegt.

Der Entwicklung und Funktionsbeschreibung der Webserver-Applikation als GUI wurde aufgrund des Umfangs ein eigenes sechstes Kapitel zugesprochen. Dort wurden Teile der Implementierung, der Aufbau der dynamischen Webseiten und die Konfigurationsmöglichkeiten der einzelnen Schnittstellentypen beschrieben.

Die Funktionsfähigkeit der Multifunktions-Adapterplatine wurde anhand der Darstellung einiger ausgewählter Tests im Kapitel 7 nachgewiesen.

Mit der Entwicklung der Multifunktions-Adapterplatine entstand ein universeller, erweiterbarer und frei konfigurierbarer Schnittstellenwandler, deren Basis eine leistungsfähige MCU bildet. Die einzelnen Kapitel dieser Masterarbeit beschreiben die einzelnen Stufen dieser Entwicklung.

## 8.2 Ausblick

Die bereits in der Phase der Konzeption eingebrachte Modularisierung durch Breakout-Sockets ermöglicht der Multifunktions-Adapterplatine auch zukünftig eine Erweiterung des Anwendungsbereiches. Durch diese Flexibilität kann dieser Schnittstellenwandler an neue Aufgabenbereiche angepasst werden.

Das Vorhandensein von fünf Datenleitungen an den für die SPI-Schnittstelle vorgesehenen Breakout-Sockets 1 und 3 ermöglicht den Betrieb von zwei SPI-Geräten pro Sockel ohne Änderung der Hardware des Base-Boards. Die hier vorhandene RESET-Leitung kann als weitere Leitung des CS-Signals genutzt werden.

Das optionale EEPROM Breakout-Board hat sich in den vergangenen Monaten als äußerst praktische Komponente zum Speichern/Laden der Konfigurationsdaten erwiesen. Das Platzieren des EEPROM-Speichers als fester Bestandteil des Base-Boards stellt eine sinnvolle und ergänzende Maßnahme zur Weiterentwicklung der Multifunktions-Adapterplatine dar.

Die oft kritisierte lange Aufbauzeit der Anzeige der dynamischen Webseiten des Webserver kann mit der Entwicklung einer eigenen GUI, beispielsweise für das Betriebssystem MS-Windows oder Linux, umgangen werden. Durch die Eigenständigkeit des Konfigurationsmoduls sind der Entwicklung weiterer Softwaremodule zur Eingabe der Konfigurationsbefehle keine Grenzen gesetzt.

Das Konzept der Multifunktions-Adapterplatine sieht nur die Aufnahme des Datenstromes der unterschiedlichsten Kommunikationsteilnehmer und das zentrale Versenden dieser Daten über die Ethernet-Schnittstelle vor. Eine softwaretechnische Erweiterung, die diesen Datenverkehr in umgekehrter Richtung realisiert, erweitert die Multifunktions-Adapterplatine zu einem vollwertigen Daten-Tunnel.

Die effiziente Ausnutzung des Speicherbedarfs der *mbed*-Software räumt viel Platz für softwaretechnische Erweiterungen ein. Mit der Version 5 des *mbed*-Softwareprojektes werden lediglich 185KiB von insgesamt 512KiB des internen MCU-Flashspeichers benutzt. Der Speicherbedarf der uneingeschränkt nutzbaren 32KiB des SRAM liegt derzeit bei 24,7KiB inklusive der speicherintensiven FIFO-Zwischenspeicher. Ein weiterer Speicherblock der Größe 16KiB wird zwar derzeit für den Aufbau der Ethernet-Datenpakete durch den uIP-Stack verwendet, dieser lässt sich bei Bedarf aber noch verkleinern. Ein weiterer 16KiB umfassender Speicherbereich, der für die Peripherie vorgesehen ist, bleibt derzeit gänzlich ungenutzt.

## A *mbed* & Breakout-Socket Mapping

| Breakout Socket 1 |                             |                                    |                     |              |
|-------------------|-----------------------------|------------------------------------|---------------------|--------------|
| Socket Pin        | <i>mbed</i> Pin / Mainboard | <i>mbed</i> 's dedicated operation | Breakout Board Type |              |
|                   |                             |                                    | General Purpose     | SPI          |
| 1                 | +3V3                        |                                    | +3V3                | +3V3         |
| 2                 | GND                         |                                    | GND                 | GND          |
| 3                 | 15                          | A/D                                | Digital I/O         | RESET / CTRL |
| 4                 | 7                           | SPI_CLK                            | Digital I/O         | SPI_CLK      |
| 5                 | 8                           | GPIO                               | Digital I/O         | GPIO / CS    |
| 6                 | 6                           | MISO                               | Digital I/O         | MISO         |
| 7                 | 5                           | MOSI                               | Digital I/O         | MOSI         |
| 8                 | +5V                         |                                    | +5V                 | +5V          |
| 9                 | GND                         |                                    | GND                 | GND          |

| Breakout Socket 2 |                             |                                    |                     |          |          |      |
|-------------------|-----------------------------|------------------------------------|---------------------|----------|----------|------|
| Socket Pin        | <i>mbed</i> Pin / Mainboard | <i>mbed</i> 's dedicated operation | Breakout Board Type |          |          |      |
|                   |                             |                                    | RS232               | RS422    | RS485    | I2C  |
| 1                 | +3V3                        |                                    | +3V3                | +3V3     | +3V3     | +3V3 |
| 2                 | GND                         |                                    | GND                 | GND      | GND      | GND  |
| 3                 | NC                          |                                    | NC                  | NC       | NC       | NC   |
| 4                 | 9                           | UART TX / I2C SDA                  | TX                  | TX       | TX       | SDA  |
| 5                 | 10                          | UART RX / I2C SCL                  | RX                  | RX       | RX       | SCL  |
| 6                 | 22                          | GPIO / PWM                         | NC                  | NC       | RTS      | GPIO |
| 7                 | 23                          | GPIO / PWM                         | DC/DC_ON            | DC/DC_ON | DC/DC_ON | GPIO |
| 8                 | +5V                         |                                    | +5V                 | +5V      | +5V      | +5V  |
| 9                 | GND                         |                                    | GND                 | GND      | GND      | GND  |

Abbildung A.1: *mbed* & Breakout-Socket 1 & 2 Mapping



## B Technische Details zum Base-Board

### B.1 Technische Spezifikation

| Spezifikation                                           | Daten                                                  |
|---------------------------------------------------------|--------------------------------------------------------|
| Abmessungen                                             | 160x100 mm (Europakarte)                               |
| Eingangsspannung                                        | 9...36 V DC (DC/DC-Wandler)<br>oder 36...57 V DC (PoE) |
| Primäre Spannungsversorgung                             | 5 V DC @ max. 2 A                                      |
| Sekundäre Spannungsversorgung                           | 3,3 V DC @ max. 3 A                                    |
| Leistungsaufnahme,<br>primär ohne aktive Schnittstellen | 156 mA (470 mW)                                        |
| Leistungsaufnahme,<br>primär, aktiv: 3xUART, SPI, ETH   | 211 mA (1,055W)                                        |
| Leistungsaufnahme CAN, primär                           | + 46mA (230 mW)                                        |

**Tabelle B.1:** Technische Daten des Base-Boards

| Kontakt | Signal  |
|---------|---------|
| 2       | CANL    |
| 6       | GND ISO |
| 7       | CANH    |

**Tabelle B.2:** Kontaktierung der D-SUB Verbindung der CAN-Schnittstelle

## C Technische Details zum RS-232 Breakout-Board

### C.1 Technische Spezifikation

| Spezifikation                 | Daten            |
|-------------------------------|------------------|
| Abmessungen                   | 35,56x35,56 mm   |
| Primäre Spannungsversorgung   | 5 V DC           |
| Sekundäre Spannungsversorgung | 3,3 V DC         |
| Leistungsaufnahme, primär     | 31,2 mA (156 mW) |
| Leistungsaufnahme, sekundär   | 1,15 mA (3,8 mW) |
| Max. Datenübertragungsrate    | 250 kBit/s       |

**Tabelle C.1:** Technische Daten des Breakout-Boards RS-232

| Signal  | JP1 → MCon    |       | JP1 → FCon    |       |
|---------|---------------|-------|---------------|-------|
|         | Wannenstecker | D-SUB | Wannenstecker | D-SUB |
| RX      | 5             | 3     | 3             | 2     |
| TX      | 3             | 2     | 5             | 3     |
| GND ISO | 9             | 5     | 9             | 5     |

**Tabelle C.2:** Kontaktierung des RS-232 Breakout-Boards

## D Konzeptionelle Details zum RS-422 Breakout-Board

### D.1 Technische Spezifikation

| Spezifikation                 | Daten          |
|-------------------------------|----------------|
| Abmessungen                   | 35,56x35,56 mm |
| Primäre Spannungsversorgung   | 5 V DC         |
| Sekundäre Spannungsversorgung | 3,3 V DC       |
| Max. Datenübertragungsrate    |                |
| <i>ADuM1201ARZ</i> :          | 2 MBit/s       |
| <i>ADuM1201BRZ</i> :          | 10 MBit/s      |

**Tabelle D.1:** Technische Daten des Breakout-Boards RS-422

| Kontakt | Printklemme | Signal  |
|---------|-------------|---------|
|         | 1           | A       |
|         | 2           | B       |
|         | 3           | Y       |
|         | 4           | Z       |
|         | 5           | GND ISO |
|         | 6           | Shield  |

**Tabelle D.2:** Kontaktierung des RS-422 Breakout-Boards

## E Konzeptionelle Details zum RS-485 Breakout-Board

### E.1 Technische Spezifikation

| Spezifikation                 | Daten          |
|-------------------------------|----------------|
| Abmessungen                   | 35,56x35,56 mm |
| Primäre Spannungsversorgung   | 5 V DC         |
| Sekundäre Spannungsversorgung | 3,3 V DC       |
| Max. Datenübertragungsrate    |                |
| ADUM 1301 ARW:                | 2 MBit/s       |
| ADUM 1301 BRW:                | 10 MBit/s      |

**Tabelle E.1:** Technische Daten des Breakout-Boards RS-485

| Kontakt | Printklemme | Signal  |
|---------|-------------|---------|
| 1       |             | A       |
| 2       |             | B       |
| 3       |             | GND ISO |
| 4       |             | Shield  |

**Tabelle E.2:** Kontaktierung des RS-485 Breakout-Boards

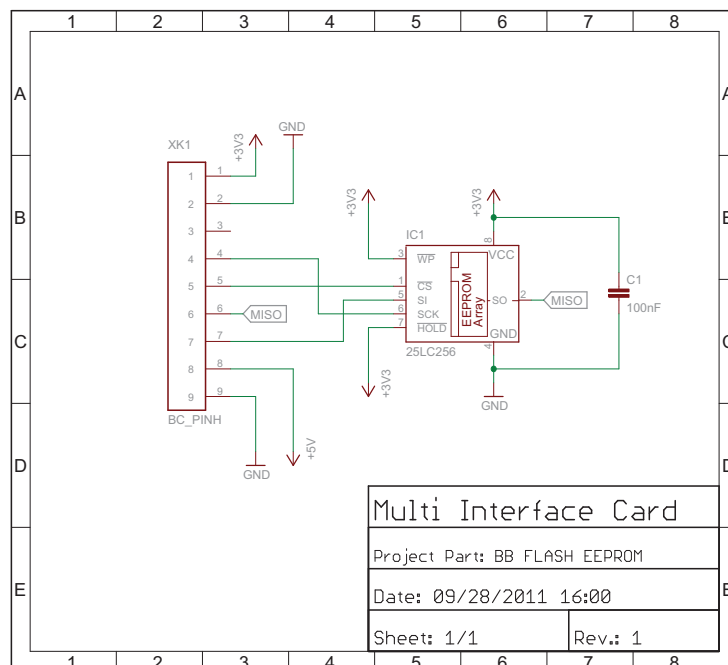
## F Technische Details zum EEPROM-Breakout-Board

### F.1 Technische Spezifikation

| Spezifikation       | Daten             |
|---------------------|-------------------|
| Abmessungen         | 35,56x35,56 mm    |
| Spannungsversorgung | 3,3 V DC          |
| EEPROM              | Microchip 25LC256 |
| Speicherkapazität   | 32768 Bytes       |
| Interface           | SPI               |

**Tabelle F.1:** Technische Daten des EEPROM-Breakout-Boards

### F.2 Schaltplan



**Abbildung F.1:** Schaltplan des EEPROM-Breakout-Boards

## G Übersicht aller Konfigurationsbefehle

### G.1 Konfigurationsbefehle der Hardwaregruppe mbed

| Action Code<br>(HEX) | Config Code<br>(DEC) | Parameter<br>(DEC) | Function                           |
|----------------------|----------------------|--------------------|------------------------------------|
| 1                    | 1                    | 0                  | ADC – Deactivate all four Channels |
|                      |                      | 1                  | ADC – Activate all four Channels   |
|                      |                      | 2                  | ADC – Select only CH1              |
|                      |                      | 3                  | ADC – Select only CH2              |
|                      |                      | 4                  | ADC – Select only CH3              |
|                      |                      | 5                  | ADC – Select only CH4              |
| 2                    | 2                    | unsigned int       | ADC Sample Rate in Hz              |
| 3                    | 3                    | unsigned char      | ID ADC                             |
| 4                    | 4                    | unsigned char      | ID BS1                             |
| 5                    | 5                    | unsigned char      | ID BS2                             |
| 6                    | 6                    | unsigned char      | ID BS3                             |
| 7                    | 7                    | unsigned char      | ID BS4                             |
| 8                    | 8                    | unsigned char      | ID CAN                             |
| 9                    | 9                    | unsigned char      | ID USB                             |
| A                    | 10                   | 0                  | ADC FIFO is in blocking Mode       |
|                      |                      | 1                  | ADC FIFO is in non blocking Mode   |
| F                    | 15                   | X                  | Initialize <i>mbed</i> Functions   |

**Tabelle G.1:** Konfigurationsbefehle der Hardwaregruppe 0h (*mbed*)

## G.2 Konfigurationsbefehle der Hardwaregruppe Breakout-Socket 1

| Action Code<br>(HEX) | Config Code<br>(DEC) | Parameter<br>(DEC) | Function                            |
|----------------------|----------------------|--------------------|-------------------------------------|
| 1                    | 17                   | 0                  | Type: Undefined                     |
|                      |                      | 4                  | Type: SPI                           |
|                      |                      | 6                  | Type: General Purpose               |
|                      |                      | 7                  | Type: Flash                         |
| 7                    | 23                   | unsigned int       | SPI CLK in Hz                       |
| 8                    | 24                   | 0                  | SPI Mode=0, CPOL=0, CPHA=0          |
|                      |                      | 1                  | SPI Mode=1, CPOL=0, CPHA=1          |
|                      |                      | 2                  | SPI Mode=2, CPOL=1, CPHA=0          |
|                      |                      | 3                  | SPI Mode=3, CPOL=1, CPHA=1          |
| 9                    | 25                   | unsigned int       | SPI Request Frequency in Hz (0=OFF) |
| A                    | 26                   | unsigned int       | SPI Commandword                     |
| B                    | 27                   | unsigned int       | SPI RESET for x*100us               |
| C                    | 28                   | X                  | SPI RESET                           |
| D                    | 29                   | 0                  | All GPIOs OFF                       |
|                      |                      | 1                  | All GPIOs ON                        |
|                      |                      | 30                 | BB1 Pin3 Output LOW                 |
|                      |                      | 31                 | BB1 Pin3 Output HIGH                |
|                      |                      | 40                 | BB1 Pin4 Output LOW                 |
|                      |                      | 41                 | BB1 Pin4 Output HIGH                |
|                      |                      | 50                 | BB1 Pin5 Output LOW                 |
|                      |                      | 51                 | BB1 Pin5 Output HIGH                |
|                      |                      | 60                 | BB1 Pin6 Output LOW                 |
|                      |                      | 61                 | BB1 Pin6 Output HIGH                |
|                      |                      | 70                 | BB1 Pin7 Output LOW                 |
|                      |                      | 71                 | BB1 Pin7 Output HIGH                |
| E                    | 30                   | 0                  | SPI FIFO is in blocking Mode        |
|                      |                      | 1                  | SPI FIFO is in non blocking Mode    |
| F                    | 31                   | X                  | Initializing and DC/DC switch ON    |

**Tabelle G.2:** Konfigurationsbefehle der Hardwaregruppe 1h (Breakout-Socket 1)

### G.3 Konfigurationsbefehle der Hardwaregruppe Breakout-Socket 2

| Action Code<br>(HEX) | Config Code<br>(DEC) | Parameter<br>(DEC) | Function                                    |
|----------------------|----------------------|--------------------|---------------------------------------------|
| 1                    | 33                   | 0                  | Type: Undefined                             |
|                      |                      | 1                  | Type: UART3 for RS-232                      |
|                      |                      | 2                  | Type: UART3 for RS-422                      |
|                      |                      | 3                  | Type: UART3 for RS-485                      |
|                      |                      | 5                  | Type: $I^2C1$                               |
| 2                    | 34                   | int                | UART: Baudrate for UART in Bps              |
| 3                    | 35                   | 5,6,7,8            | UART: Bit wordlength of a character         |
| 4                    | 36                   | 1,2                | UART: Stop-Bits                             |
| 5                    | 37                   | 0                  | UART: Parity Check disable                  |
|                      |                      | 1                  | UART: Parity Check enable                   |
| 6                    | 38                   | 0                  | UART: ODD Parity                            |
|                      |                      | 1                  | UART: EVEN Parity                           |
| 7                    | 39                   | 0                  | UART FIFO is in blocking Mode               |
|                      |                      | 1                  | UART FIFO is in non blocking Mode           |
| 8                    | 40                   | 0                  | $I^2C1$ Slave Receiver Mode                 |
|                      |                      | 1                  | $I^2C1$ Master Transmitter Mode             |
| 9                    | 41                   | 0                  | $I^2C1$ Standard Mode 100 Kbit/s            |
|                      |                      | 1                  | $I^2C1$ Fast Mode 400 Kbit/s                |
| A                    | 42                   | 7Bit               | $I^2C1$ Own address in Slave Mode           |
| B                    | 43                   | 7Bit               | $I^2C1$ Destination address for Master Mode |
| C                    | 44                   | unsigned char      | $I^2C1$ Commandbyte                         |
| D                    | 45                   | 0                  | $I^2C1$ FIFO is in blocking Mode            |
|                      |                      | 1                  | $I^2C1$ FIFO is in non blocking Mode        |
| E                    | 46                   | unsigned char      | UART FIFO Message Minimum                   |
| F                    | 47                   | X                  | Initializing and DC/DC switch ON            |

**Tabelle G.3:** Konfigurationsbefehle der Hardwaregruppe 2h (Breakout-Socket 2)

## G.4 Konfigurationsbefehle der Hardwaregruppe Breakout-Socket 3

| Action Code<br>(HEX) | Config Code<br>(DEC) | Parameter<br>(DEC) | Function                            |
|----------------------|----------------------|--------------------|-------------------------------------|
| 1                    | 49                   | 0                  | TYPE: Undefined                     |
|                      |                      | 1                  | TYPE: UART1 for RS-232              |
|                      |                      | 2                  | TYPE: UART1 for RS-422              |
|                      |                      | 3                  | TYPE: UART1 for RS-485              |
|                      |                      | 4                  | TYPE: SPI0                          |
| 2                    | 50                   | int                | UART: Baudrate for UART in Bps      |
| 3                    | 51                   | 5,6,7,8            | UART: Bit Wordlength of a character |
| 4                    | 52                   | 1,2                | UART: Stopbits                      |
| 5                    | 53                   | 0                  | UART: Parity Check disable          |
|                      |                      | 1                  | UART: Parity Check enable           |
| 6                    | 54                   | 0                  | UART: ODD Parity                    |
|                      |                      | 1                  | UART: EVEN Parity                   |
| 7                    | 55                   | unsigned int       | SPI CLK in Hz                       |
| 8                    | 56                   | 0                  | SPI Mode=0, CPOL=0, CPHA=0          |
|                      |                      | 1                  | SPI Mode=1, CPOL=0, CPHA=1          |
|                      |                      | 2                  | SPI Mode=2, CPOL=1, CPHA=0          |
|                      |                      | 3                  | SPI Mode=3, CPOL=1, CPHA=1          |
| 9                    | 57                   | unsigned int       | SPI Request Frequency in Hz (0=OFF) |
| A                    | 58                   | unsigned int       | SPI Commandbyte                     |
| B                    | 59                   | unsigned int       | SPI RESET for x*100us               |
| C                    | 60                   | X                  | SPI RESET                           |
| D                    | 61                   | 0                  | SPI FIFO is in blocking Mode        |
|                      |                      | 1                  | SPI FIFO is in non blocking Mode    |
| E                    | 62                   | unsigned char      | UART FIFO Message Minimum           |
| F                    | 63                   | X                  | IF Initializing and DC/DC switch ON |

**Tabelle G.4:** Konfigurationsbefehle der Hardwaregruppe 3h (Breakout-Socket 3)

**G.5 Konfigurationsbefehle der Hardwaregruppe Breakout-Socket 4**

| Action Code<br>(HEX) | Config Code<br>(DEC) | Parameter<br>(DEC) | Function                                    |
|----------------------|----------------------|--------------------|---------------------------------------------|
| 1                    | 65                   | 0                  | TYPE: Undefined                             |
|                      |                      | 1                  | TYPE: UART2 for RS-232                      |
|                      |                      | 2                  | TYPE: UART2 for RS-422                      |
|                      |                      | 3                  | TYPE: UART2 for RS-485                      |
|                      |                      | 5                  | TYPE: $I^2C2$                               |
| 2                    | 66                   | int                | UART: Baudrate for UART in Bps              |
| 3                    | 67                   | 5,6,7,8            | UART: Bit Wordlength of a character         |
| 4                    | 68                   | 1,2                | UART: Stopbits                              |
| 5                    | 69                   | 0                  | UART: Parity Check disable                  |
|                      |                      | 1                  | UART: Parity Check enable                   |
| 6                    | 70                   | 0                  | UART: ODD Parity                            |
|                      |                      | 1                  | UART: EVEN Parity                           |
| 7                    | 71                   | 0                  | UART FIFO is in blocking Mode               |
|                      |                      | 1                  | UART FIFO is in non blocking Mode           |
| 8                    | 72                   | 0                  | $I^2C2$ Slave Receiver Mode                 |
|                      |                      | 1                  | $I^2C2$ Master Transmitter Mode             |
| 9                    | 73                   | 0                  | $I^2C2$ Standard Mode 100 Kbit/s            |
|                      |                      | 1                  | $I^2C2$ Fast Mode 400 Kbit/s                |
| A                    | 74                   | 7Bit               | $I^2C2$ Own address in Slave Mode           |
| B                    | 75                   | 7Bit               | $I^2C2$ Destination address for Master Mode |
| C                    | 76                   | unsigned char      | $I^2C2$ Commandbyte                         |
| D                    | 77                   | 0                  | $I^2C2$ FIFO is in blocking Mode            |
|                      |                      | 1                  | $I^2C2$ FIFO is in non blocking Mode        |
| E                    | 78                   | unsigned char      | UART Message Minimum                        |
| F                    | 79                   | X                  | IF Initializing and DC/DC switch ON         |

**Tabelle G.5:** Konfigurationsbefehle der Hardwaregruppe 4h (Breakout-Socket 4)

## G.6 Konfigurationsbefehle der Hardwaregruppe CAN

| Action Code<br>(HEX) | Config Code<br>(DEC) | Parameter<br>(DEC) | Function                            |
|----------------------|----------------------|--------------------|-------------------------------------|
| 1                    | 81                   | unsigned int       | set CAN baudrate in bit per second  |
| 2                    | 82                   | 0                  | Acceptance Filter: Bypass           |
|                      |                      | 1                  | Acceptance Filter: Filter On        |
| 3                    | 83                   | unsigned short int | 1st Explizit Filter ID Standard     |
| 4                    | 84                   | unsigned short int | 2nd Explizit Filter ID Standard     |
| 5                    | 85                   | unsigned short int | 3rd Explizit Filter ID Standard     |
| 6                    | 86                   | unsigned short int | 4th Explizit Filter ID Standard     |
| 7                    | 87                   | unsigned short int | 5th Explizit Filter ID Standard     |
| 8                    | 88                   | unsigned short int | 6th Explizit Filter ID Standard     |
| 9                    | 89                   | unsigned short int | 7th Explizit Filter ID Standard     |
| A                    | 90                   | unsigned short int | 8th Explizit Filter ID Standard     |
| B                    | 91                   | 0                  | CAN FIFO is in blocking Mode        |
|                      |                      | 1                  | CAN FIFO is in non blocking Mode    |
| E                    | 94                   | X                  | De-Initialize CAN BUS Interface     |
| F                    | 95                   | X                  | IF Initializing and DC/DC switch ON |

**Tabelle G.6:** Konfigurationsbefehle der Hardwaregruppe 5h (CAN)

## G.7 Konfigurationsbefehle der Hardwaregruppe ETHERNET

| Action Code<br>(HEX) | Config Code<br>(DEC) | Parameter<br>(DEC) | Function                             |
|----------------------|----------------------|--------------------|--------------------------------------|
| 1                    | 97                   | unsigned char      | 1st Ipv4 block of own adress         |
| 2                    | 98                   | unsigned char      | 2nd Ipv4 block of own adress         |
| 3                    | 99                   | unsigned char      | 3rd Ipv4 block of own adress         |
| 4                    | 100                  | unsigned char      | 4th Ipv4 block of own adress         |
| 5                    | 101                  | unsigned char      | 1st Ipv4 block of destination adress |
| 6                    | 102                  | unsigned char      | 2nd Ipv4 block of destination adress |
| 7                    | 103                  | unsigned char      | 3rd Ipv4 block of destination adress |
| 8                    | 104                  | unsigned char      | 4th Ipv4 block of destination adress |
| 9                    | 105                  | unsigned int       | own Port (optional)                  |
| A                    | 106                  | unsigned int       | UDP Destination Port                 |
| B                    | 107                  | 0                  | UDP Broadcast: No                    |
|                      |                      | 1                  | UDP Broadcast: Yes                   |
| E                    | 110                  | X                  | Change Client address                |
| F                    | 111                  | X                  | Change Host address                  |

**Tabelle G.7:** Konfigurationsbefehle der Hardwaregruppe 6h (ETHERNET)

## G.8 Konfigurationsbefehle der Hardwaregruppe Extended ETHERNET2

| Action Code<br>(HEX) | Config Code<br>(DEC) | Parameter<br>(DEC) | Function        |
|----------------------|----------------------|--------------------|-----------------|
| 1                    | 129                  |                    | 1st Subnet Mask |
| 2                    | 130                  | unsigned char      | 2nd Subnet Mask |
| 3                    | 131                  | unsigned char      | 3rd Subnet Mask |
| 4                    | 132                  | unsigned char      | 4th Subnet Mask |
| 5                    | 133                  | unsigned char      | 1st MAC Byte    |
| 6                    | 134                  | unsigned char      | 2nd MAC Byte    |
| 7                    | 135                  | unsigned char      | 3rd MAC Byte    |
| 8                    | 136                  | unsigned char      | 4th MAC Byte    |
| 9                    | 137                  | unsigned char      | 5th MAC Byte    |
| A                    | 138                  | unsigned char      | 6th MAC Byte    |

**Tabelle G.8:** Konfigurationsbefehle der Hardwaregruppe Extended 8h (ETHERNET2)

## G.9 Konfigurationsbefehle der Hardwaregruppe Extended CAN2

| Action Code<br>(HEX) | Config Code<br>(DEC) | Parameter<br>(DEC) | Function                        |
|----------------------|----------------------|--------------------|---------------------------------|
| 1                    | 145                  | Unsigned int short | Group Filter ID Start 1 STD     |
| 2                    | 146                  | Unsigned int short | Group Filter ID End 1 STD       |
| 3                    | 147                  | Unsigned int short | Group Filter ID Start 2 STD     |
| 4                    | 148                  | Unsigned int short | Group Filter ID End 2 STD       |
| 5                    | 149                  | Unsigned int short | Group Filter ID Start 3 STD     |
| 6                    | 150                  | Unsigned int short | Group Filter ID End 3 STD       |
| 7                    | 151                  | Unsigned int short | Group Filter ID Start 4 STD     |
| 8                    | 152                  | Unsigned int short | Group Filter ID End 4 STD       |
| 9                    | 153                  | unsigned int       | 1st Explizit Filter ID Extended |
| A                    | 154                  | unsigned int       | 2nd Explizit Filter ID Extended |
| B                    | 155                  | unsigned int       | 3rd Explizit Filter ID Extended |
| C                    | 156                  | unsigned int       | 4th Explizit Filter ID Extended |
| D                    | 157                  | unsigned int       | 5th Explizit Filter ID Extended |
| E                    | 158                  | unsigned int       | 6th Explizit Filter ID Extended |
| F                    | 159                  | unsigned int       | 7th Explizit Filter ID Extended |

**Tabelle G.9:** Konfigurationsbefehle der Hardwaregruppe Extended 9h (CAN2)

## G.10 Konfigurationsbefehle der Hardwaregruppe Extended CAN3

| Action Code<br>(HEX) | Config Code<br>(DEC) | Parameter<br>(DEC) | Function                        |
|----------------------|----------------------|--------------------|---------------------------------|
| 1                    | 161                  | unsigned int       | 8th Explizit Filter ID Extended |
| 2                    | 162                  | unsigned int       | Group Filter ID Start 1 EXT     |
| 3                    | 163                  | unsigned int       | Group Filter ID End 1 EXT       |
| 4                    | 164                  | unsigned int       | Group Filter ID Start 2 EXT     |
| 5                    | 165                  | unsigned int       | Group Filter ID End 2 EXT       |
| 6                    | 166                  | unsigned int       | Group Filter ID Start 3 EXT     |
| 7                    | 167                  | unsigned int       | Group Filter ID End 3 EXT       |
| 8                    | 168                  | unsigned int       | Group Filter ID Start 4 EXT     |
| 9                    | 169                  | unsigned int       | Group Filter ID End 4 EXT       |

**Tabelle G.10:** Konfigurationsbefehle der Hardwaregruppe Extended Ah (CAN3)

## G.11 Konfigurationsbefehle der Hardwaregruppe Extended Others

| Action Code<br>(HEX) | Config Code<br>(DEC) | Parameter<br>(DEC) | Function                                |
|----------------------|----------------------|--------------------|-----------------------------------------|
| 1                    | 177                  | 0                  | BB3: UART1 FIFO is in blocking Mode     |
|                      |                      | 1                  | BB3: UART1 FIFO is in non blocking Mode |
| 2                    | 178                  | unsigned int       | BB2 UART FIFO Release Time              |
| 3                    | 179                  | unsigned int       | BB3 UART FIFO Release Time              |
| 4                    | 180                  | unsigned int       | BB4 UART FIFO Release Time              |

**Tabelle G.11:** Konfigurationsbefehle der Hardwaregruppe Extended Bh (Others)

## Literaturverzeichnis

- [1] Universal serial bus specification revision 2.0, 2000.
- [2] Guidelines for proper wiring of an rs-485 (tia/eia-485-a) network. Application Note AN763, MAXIM, 2001.
- [3] Rs-485 (eia/tia-485) differential data transmission system basics. Application Note AN736, MAXIM, 2001.
- [4] Tvs diodes - Überspannungs-schutzdioden. Technical report, Diotec Semiconductor GmbH, 2001.
- [5] Tvs diode application note. Application Note AN96-07, SEMTECH International AG, 2002.
- [6] I2c-bus specification and user manual v.3.0. User manual UM10204, NXP, 2007.
- [7] Interfacing to high speed adcs via spi. Application Note AN877, Analog Devices, 2007.
- [8] Ronn Kliger Baoxing Chen, John Wynne. High speed digital isolators using microscale on-chip transformers. *Elektronik*, 15:38–42, 2003.
- [9] Sam Broyles. A system evaluation of can transceivers. Application Report SLLA109, Texas Instruments, 2002.
- [10] Manny Soltero Jing Zhang Chris Cockril. Rs-422 and rs-485 standards overview and system configurations. Application Report SLLA070D, Texas Instruments, 2010.
- [11] Steve Corrigan. Introduction to the controller area network (can). Application Report SLOA101A, Texas Instruments, 2008.
- [12] Dieter Conrads Franz Demmelmeier. *Serielle Busse*. vde-Verlag GmbH, 1987.
- [13] Edwin Wright Deon Reynders. *Practical TCP/IP and Ethernet Networking for Industry*. Elsevier Ltd., 2003.
- [14] Adam Dunkels. *The uIP Embedded TCP/IP Stack*. Swedish Institute of Computer Science, the uip 1.0 reference manual edition, June 2006.
- [15] Joachim Franz. *EMV - Störungssicherer Aufbau elektronischer Schaltungen*. Vieweg+Teubner Verlag, 2011.

- [16] Martin Horneffer Kristof Obermann. *Datennetztechnologien für Next Generation Networks: Ethernet, IP, MPLS und andere*. Vieweg+Teubner, 2009.
- [17] Tony Morgan. Power over ethernet - the reality of designing a powered device, 2006.
- [18] L.H. Mweene. Power supply design for power-over-ethernet applications. Power Designer 104, National Semiconductor, 2005.
- [19] NXP. *LPC17xx User manual*, 08 2010.
- [20] Harald Schumny Rainer Ohl. *Handbuch Digitaler Schnittstellen*. Vieweg, 1994.
- [21] Joe Vo. A comparison of differential termination techniques. Application Note AN903, National Semiconductor, 1993.
- [22] Michael R. Wilson. Tia/eia-422-b overview. Application Note AN1031, National Semiconductor, 2000.
- [23] Friedrich Wittgruber. *Digitale Schnittstellen und Bussysteme*. Vieweg, 2002.

## **Erklärung über die selbständige Anfertigung der Arbeit**

Ich versichere, dass ich die vorliegende Arbeit selbständig und ohne unerlaubte Hilfe Dritter verfasst sowie keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Angaben, die inhaltlich oder wörtlich aus fremden Werken stammen, wurden kenntlich gemacht. Diese Arbeit lag in gleicher oder ähnlicher Weise noch keiner Prüfungsbehörde, Fachhoch- oder Hochschule vor und wurde ebenfalls bisher noch nicht veröffentlicht.

Rostock, 14.01.2012

Maik Gotzmann

## Datenträger

Der Datenträger zu dieser Masterarbeit enthält:

- Masterarbeit im Format PDF und LaTeX
- EAGLE Projekte und Bauteilbibliothek
- *mbed*-Softwareprojekt
- PERL-Testscript