Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

DIPLOMARBEIT

Entwicklung einer grafischen Parametrier- und Visualisierungsumgebung für Umrichter unter Verwendung von MATLAB

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs unter der Leitung von

O.Univ.Prof. Dipl.-Ing. Dr.techn. Manfred Schrödl

und Betreuung durch

Dipl.-Ing. Mag.rer.soc.oec. Daniel Josef Asch

sowie

Dipl.-Ing. Wolfgang Staffler

E372
Institut für elektrische Antriebe und Maschinen

eingereicht an der technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von

Werner Kloiber, BSc.

Matrikelnummer 0125856
Tiefenbachstraße 18a, 5300 Hallwang

Wien, November 2010

Danksagung

Durchhalten! Dieses Verb ist für mich die wohl beste Beschreibung, wie das Studium der Elektrotechnik absolvierbar ist.
Um durchhalten zu können, bedarf es aber auch Unterstützung aus dem Umfeld. Sei es von Professoren und Assistenten, die Fragen stets geduldig und ausführlich beantworten, von Freunden, die immer mit einem offenen Ohr beiseite stehen und mit einem durch dick und dünn gehen, aber auch von der Familie, die einem den nötigen Halt und die moralische Unterstützung bieten.
Mein besonderer Dank gilt in diesem Sinn Prof. Schrödl, der es mir ermöglichte, an seinem Institut diese Diplomarbeit zu erstellen, sowie DI Mag. Daniel Asch und DI Wolfgang Staffler, die mich bei meiner Diplomarbeit ausgesprochen gut und stets geduldig betreut haben.
Weiters möchte ich mich auf diesem Weg bei meinen Freunden bedanken, auf deren Unterstützung man sich stets verlassen konnte und die auch für die nötige Zerstreuung sorgten. Ganz speziell gilt mein Dank Reinhard, der sich für das Korrekturlesen dieser Arbeit Zeit genommen hat.
Zu meinem Studienerfolg hat ganz wesentlich meine Partnerin Isabella beigetragen, die mir trotz meiner Launen, die das Studium so mit sich bringt, immer gut zugesprochen hat. Dafür gilt ihr mein ganz besonderer Dank!
Zu guter Letzt gilt mein Dank meinen Eltern, die meine Entscheidung für dieses Studium stets befürwortet und mich unterstützt haben.

Abstract

For a field oriented control (FOC) off asynchronous, synchronous and permanent-magnetic synchronous motors it is necessary to use inverters. These inverters have to be adjusted accordingly by there parameters to run the specific machine.
Theme of this diploma project was to create a MATLAB based graphical user interface for adjusting the parameters of an inverter, which is able to communicate with the inverter via a serial, a network or an internet connection. The big advantage of using an internet connection in this context is the possibility to adjust the parameters over a big distance.
The MATLAB-GUI is able to read, to list clear and to manipulate the parameters. Further it is possible to show the received data on a chart, called “The Eye”. The GUI is also able to show structure elements of parameters and, if necessary, can administrate parameters in groups.
The big advantages of using MATLAB are the possibility of mathematical treatment of the received data and the option to increase the functionality of the MATLAB-GUI in a simple way.
This document describes the requirements to the Project and shows a short overview of the Hardware. Further it describes the communication protocol and every individual telegram.
The detailed explanation of the MATLAB-GUI is the most essential part of this document. This part explains the requirements of the GUI, the start of a project and all functions in the GUI. Additionally there is a detailed explanation of “The Eye”, which is used for a graphical presentation of the received data. Further it give details about changing the software parameter and how the software can be protected against manipulation.
The last chapter in this document shows an overview of the most important software functions, which are ordered by the most significant assignments. This overview should help to service or to upgrade the software.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

ASM	Asynchronmaschine
bzw.	beziehungsweise
DSP	Digitaler Signalprozessor
empf.	empfangen
FOC	Field Oriented Control
GUI	Graphical User Interface
IEAM	Institut für elektrische Antriebe und Maschinen
LSB	Least Significant Bit
MCR	Matlab Compiler Runtime
MSB	Most Significant Bit
PC	Personal Computer
PSM	Permanenterregte Synchronmaschine
RAM	Random-Access-Memory
SCI	Serial Communication Interface
SM	Synchronmaschine
TCP/IP	Transmission Control Protocol / Internet Protocol
u.a.	unter anderem
usw.	und so weiter
z.B.	zum Beispiel

1. Einleitung

1.1 Umrichter

Asynchronmaschinen (ASM) finden in der elektrischen Antriebstechnik eine starke Verbreitung. Sie sind robust und preisgünstig und können direkt am Netz betrieben werden. Nachteilig ist jedoch, dass die Drehzahl nach $n \approx n_1 = f_1 / p$ eng an die Frequenz (f) der Ständerspannung und die Polpaarzahl (p) gebunden ist. Im normalen Netzbetrieb (50 Hz) sind somit, abhängig von der Polpaarzahl, nur Drehzahlen um 3000 U/min, 1500 U/min, 1000 U/min usw. erreichbar[1].
Die mittlerweile hoch entwickelte Leistungselektronik erlaubt es jedoch, Umrichter zu bauen, die eine effiziente Regelung von elektrischen Maschinen ermöglichen.
Von wesentlicher Bedeutung ist dabei die feldorientierte Regelung von permanent-erregten Synchronmaschinen (PSM). Dieser Maschinentyp weist, wie in Abbildung 1.1 erkennbar, einen sehr hohen Wirkungsgrad auf. Aufgrund des hohen Wirkungsgrades und der mittlerweile sehr leistungsfähigen Umrichter, die für den Betrieb einer PSM notwendig sind, finden diese Motoren in der elektrischen Antriebstechnik immer häufiger Verwendung.

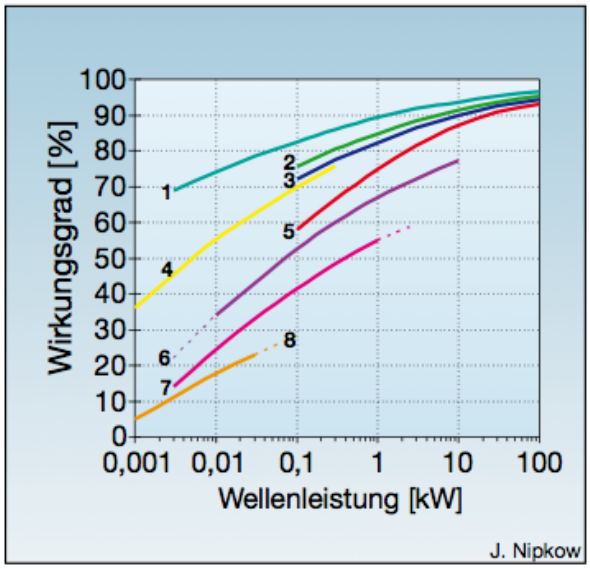


Abbildung 1.1[2]: Wirkungsgrade verschiedener Motoren kleiner Leistung
1 Permanentmagnetmotor spezial, elektronisch
*2 Asynchronmotor, 3-Stern ****
*3 Asynchronmotor, 2-Stern ***
4 Gleichstrom Permanentmagnetmotor, konventionell
*5 Asynchronmotor, 1-Stern **
6 Einphasiger Asynchronmotor, Betriebskondensator
7 Universalmotor (Kollektormotor)
8 Einphasiger Spaltpol-Asynchronmotor
**** (3-Stern), ** (2-Stern) und * (1-Stern) bezeichnet die Energieeffizienzklasse der ASM*

[1] Vergleich: Fischer, Rolf: (Elektrische Maschinen), S. 180
[2] Quelle: Nipkow, Jürg: (Energieeffizienz bei Elektromotoren), S. 16

Umrichter setzen sich aus einem Leistungsteil und einem Steuerungsteil zusammen. Der Leistungsteil besteht aus zwei in Reihe geschalteten Stromrichtern, welche über einen Zwischenkreis miteinander verbunden sind. Der Zwischenkreis beinhaltet einen oder mehrere Energiespeicher um die beiden Teilstromrichter voneinander zu entkoppeln.
Im Motorbetrieb arbeitet der erste Teilstromrichter als Gleichrichter und der zweite Teilstromrichter als Wechselrichter. Im Generatorbetrieb kehren die beiden Stromrichter ihre Betriebsart um.
Die größte praktische Bedeutung haben Umrichter mit Spannungs- bzw. mit Stromzwischenkreis. Bei Ersterem wird der zweite Stromrichter über eine eingeprägte Gleichspannung gespeist. Bei Letzterem wird Teilstromrichter 2 mit einem eingeprägtem Gleichstrom versorgt. Abbildung 1.2 zeigt die Blockschaltbilder eines Umrichters mit einem Spannungs- und einem Stromzwischenkreis.

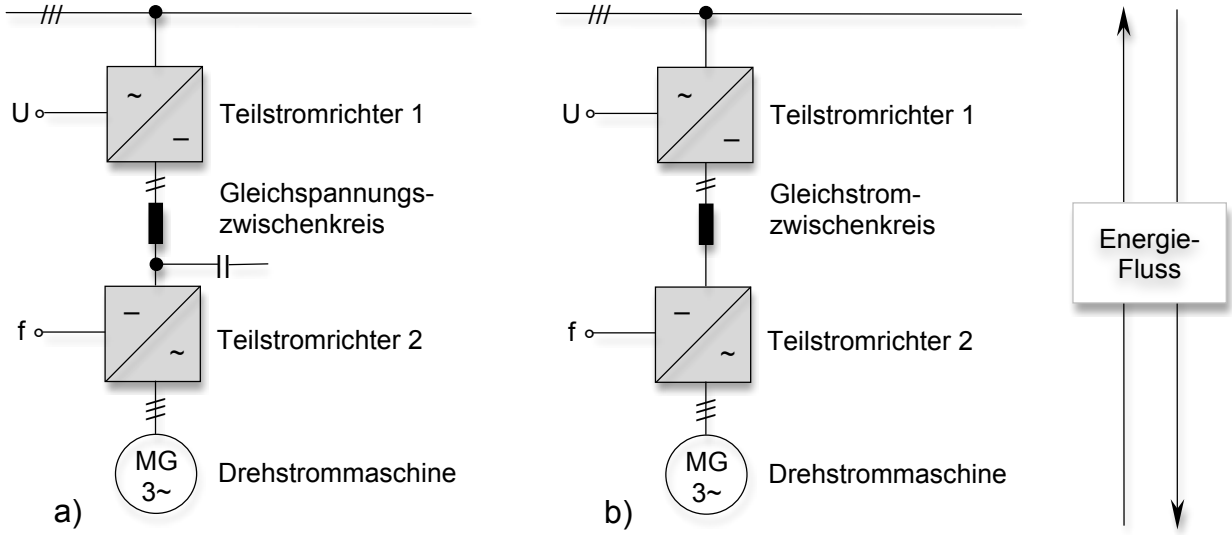


Abbildung 1.2[1]: Blockschaltbilder von Zwischenkreisumrichtern
a) mit Gleichspannungszwischenkreis
b) mit Gleichstromzwischenkreis

Der Steuerungsteil besteht aus einem Prozessor sowie aus mehreren Schnittstellen, über die alle relevanten Größen eingelesen und die elektronischen Schaltelemente des Leistungsteils angesteuert werden.
Die Auswertung der eingelesenen Größen und die entsprechende Ansteuerung der elektronischen Schaltelemente übernimmt die Firmware des Steuerungsteils. Diese muss für die jeweilige Anwendung des elektrischen Antriebes erstellt bzw. parametriert werden.

[1] Quelle: Grafik nach Bystron, Klaus: (Leistungselektronik), S. 4

1.2 Aufgabenstellung

Wie bereits unter Abschnitt 1.1 erläutert, besteht ein Umrichter aus einem Steuerungsteil und einem Leistungsteil. Der Steuerungsteil ist mit einer Firmware versehen, deren Parameter für eine effiziente Funktionsweise des Umrichters je nach Anwendung bzw. Motor angepasst werden müssen.
Am Institut für Elektrische Antriebe und Maschinen an der TU-Wien wird dieser Steuerungsteil durch ein Digital-Signal-Prozessor-Board (DSP-Board) realisiert, das eine Reihe von Schnittstellen und Funktionen bietet.
Dieses Board ermöglicht u.a. eine serielle Verbindung zu einem Computer, über den mit einer entsprechenden Software und mit einem passenden Map-File die Parametrierung der Firmware vorgenommen werden kann.
Das Map-File wird beim Kompilieren der Firmware generiert und enthält eine Auflistung aller Variablen bzw. Parameter der Firmware inklusive ihrer zugehörigen Adressen im Prozessor. Es stellt somit einen „Wegweiser" für die Parametrierungssoftware dar.

Um die Bearbeitungs- und Visualisierungsmöglichkeiten der Parameter eines Umrichters zu erweitern, soll im Zuge dieser Diplomarbeit, unter Verwendung von MATLAB[1], eine Parametrierungssoftware mit grafischer Benutzeroberfläche erstellt werden.
MATLAB ist eine plattformunabhänige Software der Firma „The MathWorks" zur Lösung von mathematischen Problemstellungen und zur grafischen Darstellung der Ergebnisse. Durch die Verknüpfung von MATLAB mit der DSP-Kommunikationssoftware entsteht somit die Möglichkeit, empfangene Messwerte direkt zu bearbeiten und grafisch darzustellen, ohne die Daten in ein weiters Programm exportieren zu müssen.
Des Weiteren soll mit der Neugestaltung des „Graphical User Interface" (GUI), neben einer seriellen Verbindung, eine TCP/IP-Verbindung möglich sein. Damit ist man in der Lage, über eine bestehende Internetverbindung eine Kommunikation zwischen DSP-Board und Computer herzustellen. Durch diese Art der Verbindung erreicht man eine räumliche Trennung zwischen den Kommunikationspartnern, was zu einer Erhöhung der Flexibilität bei der Parametrierung und bei der Inbetriebnahme von Maschinen führt.
Da es sich bei den zur Verfügung stehenden Computern um Mac- (Unix) und Windows-Plattformen handelt, soll die Kommunikationssoftware auf beiden Systemen einsetzbar sein.
Um den Verlauf der empfangenen Daten grafisch darstellen zu können, muss im Zuge der Diplomarbeit in der Benutzeroberfläche die Funktion „The Eye" integriert werden. Dadurch wird es möglich, die empfangenen Daten mit einem beliebigen Gewichtungsfaktor zu versehen, Schwellwerte zu definieren und in einer Log-Datei zu sichern um diese zu einem späteren Zeitpunkt wieder grafisch darstellen oder bearbeiten zu können.
Für die exakte Darstellung eines Signalverlaufs bietet das DSP-Board vier analoge Kanäle zum Anschluss eines Oszilloskops. Diese Funktion wird als „Softauge" bezeichnet. Mit Hilfe des GUI soll es möglich sein, dem gewünschten Kanal eine Variable inklusive Skalierungsfaktor (Zoomwert) zuzuweisen.

---

[1] Eine hilfreiche Anleitung zum Einstieg in MATLAB bietet das Buch „MATLAB-Simulink-Stateflow" von Angermann, Anne et al.

Die in der bereits existierenden Kommunikationssoftware vorhandenen Funktionen, wie die Möglichkeit der Speicherung des RAM-Inhalts, der normierten, hexadezimalen und dezimalen Darstellung der gesendeten und empfangenen Daten und die Auflistung der Variablennamen, sollen in das neue MATLAB-GUI übernommen und durch zusätzliche Funktionen ergänzt werden. Durch diese zusätzlichen Funktionen ergeben sich die Möglichkeiten, Gruppen mit unterschiedlichen Variablenlisten anzulegen sowie im Map-File aufgelistete Variablen bzw. Parameter als Struktur zu verwalten.
Um künftigen Änderungen und Erweiterungen in der verwendeten Hardware bzw. im Funktionsumfang der Kommunikationssoftware flexibel gegenüber zu stehen bzw. den Code wartbar zu gestalten, muss die Kommunikationssoftware möglichst modular in Funktionsblöcken aufgebaut werden und in der Ausführung der Programmierung leicht nachvollziehbar sein.

2. Schnittstelle zwischen Umrichter und Computer

2.1 Hardware

2.1.1 DSP-Board

Wie bereits in Abschnitt 1.2 erläutert, wird am Institut für elektrische Antriebe und Maschinen der Steuerungsteil eines Umrichters anhand eines DSP-Boards realisiert. Dieses Board, dargestellt in Abbildung 2.1, bietet ein breites Spektrum an Schnittstellen und Funktionen, die speziell für die Entwicklung der Firmware von Bedeutung sind. Falls der Umrichter nach Abschluss der Entwicklungsphase an einen Kunden ausgegeben oder in Serie produziert werden soll, wird das DSP-Board durch eine spezifische Steuereinheit ersetzt. Diese Steuereinheit wird dabei auf die für den Betrieb des Umrichters notwendigen Schnittstellen und Funktionen beschränkt.

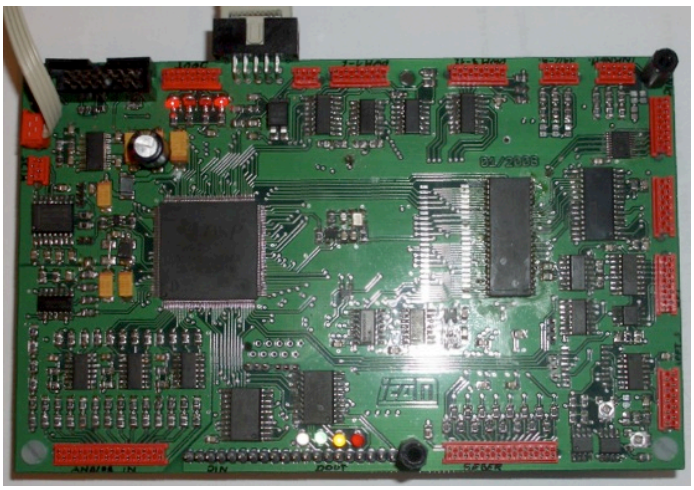


Abbildung 2.1[1]*: DSP-Board zur Steuerung eines Umrichters*

Das DSP-Board stellt u.a. ein Interface zur Verfügung, an der die vier analogen Signale der Softauge-Kanäle ausgegeben werden. An diesem Interface kann ein Adapter für vier BNC-Anschlüsse angebracht werden, durch die eine einfache Anbindung eines Oszilloskops zur Darstellung der Softauge-Variablen möglich ist.
Die Kommunikation zwischen Parametrierungssoftware und DSP erfolgt über eine serielle Verbindung und ein dafür entwickeltes Protokoll (siehe Abschnitt 2.2). Als Schnittstelle dient dem DSP dafür ein SCI-Interface.
Dieses Interface ist in Abbildung 2.1 im linken oberen Bereich zu finden, wo es ein schmales Flachbandkabel trägt. Dieses Kabel führt zum in Abschnitt 2.1.2 beschriebenen SCI-Ethernet-Adapter.
Nähere Details zum DSP-Board sind dessen Dokumentation[2] zu entnehmen.

[1] Quelle: Foto erstellt von Werner Kloiber, am 4.10.2010
[2] Eine Dokumentation bzw. nähere Informationen zum DSP-Board erhält man am Institut für elektrische Antriebe und Maschinen.

2.1.2 SCI-Ethernet-Adapter

Um die Kommunikationsmöglichkeiten zu erweitern, kann an die SCI-Schnittstelle ein SCI-Ethernet-Adapter angeschlossen werden. Der Adapter (siehe Abbildung 2.2) dient dabei als Webserver, der über die ihm fix zugewiesene IP-Adresse via TCP/IP-Protokoll erreicht werden kann.

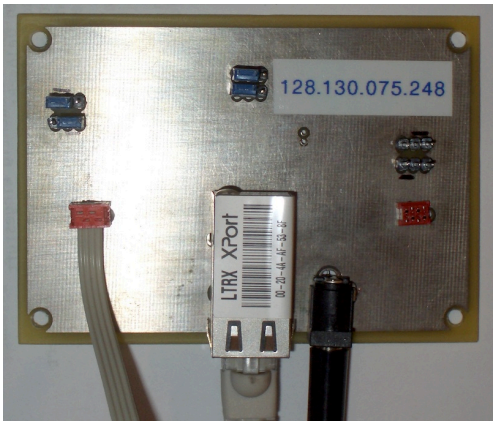


Abbildung 2.2[1]: SCI-Ethernet-Adapter
In der Mitte der Platine befindet sich ein Jumper, der die zu verwendende SCI-Schnittstelle bestimmt. Die linke Schnittstelle führt ihren Datentransfer über Optokoppler aus, die rechte Schnittstelle nutzt zum Datentransfer einen MAX232 Pegelwandler. Die Jumper oberhalb der SCI-Schnittstellen dienen dem Auskreuzen von RX und TX.

Durch die Verwendung des Webservers erfolgt die Verbindung zwischen Computer und DSP-Board über eine lokale Netzwerk- bzw. über eine Internetanbindung. Für diese Anbindung muss der Ethernet-Anschluss des Adapters mit einem Switch, Router oder Modem verbunden werden.
Sind der Umrichter und der Parametrierungscomputer über ein lokales Netzwerk oder eine Internetanbindung miteinander gekoppelt, ergibt sich der große Vorteil, dass beide voneinander räumlich getrennt betrieben werden können!
Falls erforderlich, ist es jedoch auch möglich, die Verbindung über eine direkte Ethernetverkabelung herzustellen. Um den Adapter letztlich zur Kommunikation verwenden zu können, muss die IP-Adresse sowie der für die Kommunikation vorgesehene Port bekannt sein.
Nähere Details zum SCI-Ethernet-Adapter sind dessen Dokumentation[2] zu entnehmen.

---

[1] Quelle: Foto erstellt von Werner Kloiber, am 4.10.2010
[2] Eine Dokumentation bzw. nähere Informationen zum SCI-Ethernet-Adapter erhält man am Institut für elektrische Antriebe und Maschinen.

2.2 Protokoll

2.2.1 Allgemein

Das Protokoll zur seriellen Kommunikation zwischen Computer und DSP-Board über die SCI-Schnittstelle wurde von DI Mathias Hofer und DI Wolfgang Staffler am Institut für Elektrische Antriebe und Maschinen entwickelt.
Es ist nach dem Master&Slave-Prinzip aufgebaut und beschreibt wie die einzelnen zu sendenden Telegramme zusammengestellt werden müssen. Die Rolle des Masters übernimmt der Computer. Eine Kommunikation bzw. ein Datenaustausch zwischen Computer und DSP findet somit nur statt, wenn dies vom Master angefordert wird.

Der prinzipielle Aufbau des Telegramms erfolgt folgendermaßen:

Byte 1:	Identification-Byte (entspricht immer dem dezimalen Wert 85)	
Byte 2:	Gesamtanzahl der Telegramm-Bytes (inklusive Prüfsumme)	
Byte 3:	Kommando (dient der Unterscheidung der Telegrammtypen)	
Byte 4:	Anzahl der Datenbytes	
Byte 5:	Datentyp der jeweiligen Variablen	
Byte 6:	Datenbyte 1	Datenbytes
Byte 7:	Datenbyte 2	
Byte...:	Datenbyte ...	
...	...	
...	...	
Byte x:	Prüfsumme (zur Überprüfung der Datenintegrität)	

Byte 1 dient dem Empfänger dazu, den Beginn eines Telegramms, das wie oben beschrieben aus einer Reihenfolge von mehreren unterschiedlichen Bytes besteht, zu erkennen. Der Wert des ersten Bytes wurde dafür auf dezimal 85 bzw. binär 01010101 definiert.

2.2.2 Telegrammtypen

Die Kommunikation zwischen DSP und Computer dient nicht nur der Übertragung von Variablen- bzw. Parameterwerten, sondern wird unter anderem auch zur Initialisierung des DSP oder zum Auslesen des RAM-Inhalts verwendet. Dadurch ist es notwendig, unterschiedliche Telgrammtypen festzulegen.
Um dem Empfänger mitzuteilen, um welchen Telgrammtyp es sich handelt, müssen im Protokoll Kommandos definiert werden, anhand derer der Empfänger die empfangenen Daten entsprechend verarbeiten kann. Diese Befehle werden in Byte 3 des Telegramms übermittelt.
Die definierten Kommandos sind in Tabelle 2.1 aufgelistet.

Hex-Wert	Dezimal-Wert	Befehl
0x00	0	Initialisierung
0x10	16	Schreiben von Adressen
0x20	32	Schreiben von Variablen
0x30	48	Lesen von Variablen
0x40	64	Schreiben von Softauge-Adressen
0x50	80	Schreiben von Zoomfaktoren
0x60	96	DSP-RAM auslesen
0x70	112	DSP-EEPROM auslesen

Tabelle 2.1: Kommandos zur Unterscheidung der Telegrammtypen

2.2.3 Datentypen

Die in diesem Protokoll verwendeten Datentypen beziehen sich auf einen DSP vom Typ TMS320F2xxx von Texas Instruments und sollen in diesem Abschnitt kurz erläutert werden.
Der verwendete DSP kann Daten als Long- oder Integer-Wert verarbeiten. Bei einem Long-Wert handelt es sich um eine binäre Zahl, die sich aus bis zu 32 Bit (vier Byte) zusammensetzen kann. Mit diesem Datentyp ist man in der Lage, einen Wertebereich von

$$0 \text{ bis } +\left(2^{32}-1\right) \qquad \text{... ohne Berücksichtigung eines Vorzeichens}$$

bzw.

$$-\left(2^{32}/2\right) \text{ bis } +\left(2^{32}/2-1\right) \qquad \text{... mit Berücksichtigung eines Vorzeichens}$$

darzustellen.

Bei einem Integer-Wert handelt es sich um eine binäre Zahl, die sich aus 16 Bit (zwei Byte) zusammensetzen kann. Mit diesem Datentyp ist man in der Lage, einen Wertebereich von

$$0 \text{ bis } +\left(2^{16}-1\right) \qquad \text{... ohne Berücksichtigung eines Vorzeichens}$$

bzw.

$$-\left(2^{16}/2\right) \text{ bis } +\left(2^{16}/2-1\right) \qquad \text{... mit Berücksichtigung eines Vorzeichens}$$

darzustellen.

Ob es sich bei den im Telegramm enthaltenen Datenbytes um Long- oder Integer-Datentypen handelt, wird dem Empfänger über Byte 5 der Nachricht mitgeteilt. Jedes Bit von Byte 5 steht stellvertretend für den Datentyp eines Datenbytes. Ein Bit mit dem Wert 0 signalisiert einen Integer-Datentyp, während ein Wert von 1 repräsentativ für Daten vom Typ Long steht. Dabei korrespondiert die Position des Bits im Byte mit der Reihenfolge der zu empfangenden Datenbytes.

Wenn beispielsweise Byte 5 dem binären Wert 00000101 bzw. dem dezimalen Wert 5 entspricht, bedeutet dies, dass die erste und die dritte zu empfangende Variable vom Typ Long sind und die restlichen Variablen dem Typ Integer entsprechen.
Wird im Telegramm nur eine Variable übertragen, wird Byte 5 entsprechend dem Datentyp der Wert 0 (Integer) oder 1 (Long) zugewiesen.
Der passende Datentyp jeder Variable, abhängig vom Anwendungsbereich, wird in der Firmware des DSP-Board definiert. Um die Variablen korrekt im MATLAB-GUI empfangen und darzustellen zu können, muss man sich über die definierten Datentypen der jeweiligen Größen im Klaren sein.

2.2.4 Reihenfolge der Datenbytes

Da die Nutzdaten, wie oben beschrieben, je nach Datentyp aus bis zu 32 Bits (4 Bytes) bestehen können, im Telegramm Informationen aber nur byteweise übertragen werden, müssen diese in einzelne Bytes aufgeteilt und beginnend mit dem Low-Byte (Byte 1) nacheinander gesendet werden.
Abbildung 2.3 zeigt den Aufbau und die Aufteilung einer Long- bzw. Integer-Variable.

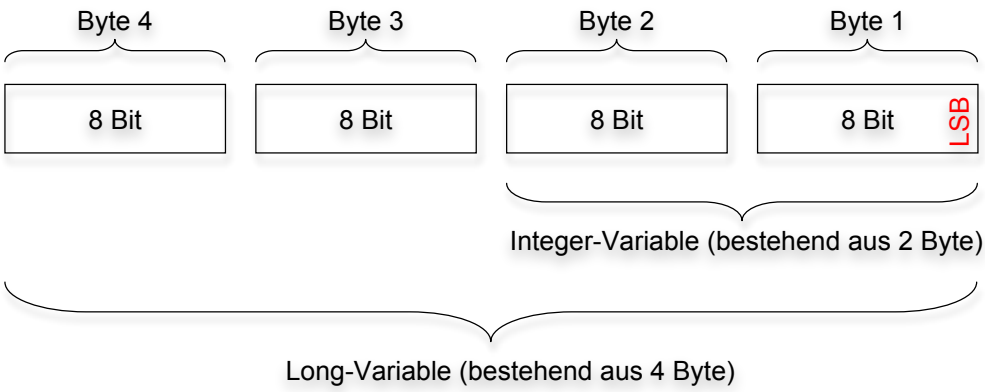


Abbildung 2.3: Aufteilung einer Long- bzw. einer Integer-Variable
Byte 1 wird zuerst gesendet

Beispielsweise handelt es sich bei dem dezimalen Wert 2480 um eine Zahl, die als Integer-Typ mit zwei Byte dargestellt werden kann:

⇒ 2480 dezimal = 00001001 10110000 binär

Das Low-Byte (Byte 1) hat in diesem Fall den binären Wert 10110000 (dezimal 176), das High-Byte (Byte 2) den binären Wert 00001001 (dezimal 9).
Diese beiden Bytes können, da sie in den Datenbereich eines Bytes fallen, problemlos übertragen werden.
Der Empfänger des Telegramms setzt die einzelnen Bytes der Nutzdaten nach dem selben Schema, nur in umgekehrter Form, wieder zu einer gesamten Zahl zusammen und kann diese im Anschluss normal weiterverarbeiten.

2.2.5 Adressierung der Variablen

Jeder Variable bzw. jedem Parameter des Umrichters ist eine Adresse zugeordnet, die dem Speicherort im RAM des DSP entspricht.
Das Auslesen oder Schreiben von Variablenwerten erfolgt jedoch nicht direkt über die Adresse der Variablen, sondern indirekt über einen Index. Dieser verweist auf einen Registereintrag im DSP, in dem die Adresse der Variable zuvor hinterlegt worden ist.
Dies setzt voraus, dass zuerst die Adressen aller Variablen, deren Werte von Interesse sind, in dem bereits erwähnten Register des DSP abgelegt werden.
Dazu muss für jede Variable ein Telegramm mit dem Kommando 0x10 (Schreiben von Adressen) an den DSP gesendet werden. Dieses enthält neben der Adresse einen Index mit einem Wert zwischen 0 und 255. Der Index entspricht jener Position, an der die Variablenadresse im Register abgelegt werden soll.
Ist das Register einmal erstellt worden, kann mit dem Index der betreffende Variablenwert vom Master jederzeit gelesen oder geschrieben werden. Dies setzt jedoch voraus, dass die Verknüpfung zwischen Variable und jeweiligem Index in der Kommunikationssoftware des Computers verwaltet wird.

2.2.6 Überprüfung der Datenintegrität

Zur Kontrolle der Datenintegrität wird am Ende eines Telegramms ein Prüfsummenbyte angefügt.

Dieses wird folgendermaßen berechnet:

255	(maximaler Wert eines Bytes)
− Summe aller Telegrammbytes	(Prüfsummenbyte nicht enthalten)
= Prüfsumme	(wenn Summe < 0 entspricht diese noch nicht dem Prüfsummenbyte)

Mit einem Byte kann, ohne Berücksichtigung eines Vorzeichens, ein Wertebereich von 0 bis 255 dargestellt werden. Da die nach dem oben dargestellten Schema berechnete Prüfsumme negativ sein kann, muss zu dieser gegebenenfalls mehrmals 256 hinzu addiert werden, bis sich ein Wert im darstellbaren Bereich ergibt. Dieser Wert zwischen 0 und 255 entspricht dem Prüfsummenbyte und wird dem Telegramm am Ende angefügt.
Sollen nun Daten vom DSP ausgelesen oder an diesen gesendet werden, wird vom Master ein Telegramm mit entsprechendem Kommando erstellt, dieses mit der berechneten Prüfsumme ergänzt und anschließend an den DSP versendet. Der Empfänger bildet beim Empfang der Daten, nach selbigem Schema wie oben erläutert, eine Prüfsumme aus der empfangenen Nachricht. Ist dabei eine Differenz zwischen der berechneten und der im Telegramm enthaltenen Prüfsumme festzustellen, wird vom DSP der Empfang der Daten nicht bestätigt. Ist nun eine Antwort des Empfängers nicht innerhalb einer vorgegebenen Zeit (Timeout) beim Absender eingetroffen, sendet der Master das Telegramm erneut.
Die Antwort des DSP enthält, neben den gegebenenfalls angeforderten Daten, ebenfalls eine Prüfsumme. Diese dient dem Master, im gleichen Sinn wie dem Slave, zur Kontrolle der empfangenen Daten. Wird eine Unstimmigkeit bei der Prüfsumme erkannt, sendet der Master, gleichfalls wie beim Ausbleiben einer Antwort, das

ursprüngliche Telegramm erneut an den Empfänger. Dies führt bei korrekter Übertragung and den DSP zu einer neuerlichen Bestätigung, deren Integrität abermals kontrolliert wird.
Erst wenn die Kontrolle der Prüfsumme kein Problem aufzeigt, werden die empfangenen Daten vom Master bzw. vom Slave verarbeitet.
Im Falle einer längerfristigen Störung des Übertragungsweges, würde der Master das Senden des Telegramms aus den oben genannten Gründen immerzu wiederholen. Um das abermalige Senden der Nachricht auf einige Wiederholungen zu beschränken, ist ein Zeitlimit in der Software des Absenders definiert (siehe Tabelle 2.4, Variable „Checksum_timeout"). Innerhalb dieses Zeitlimits muss der fehlerfreie Kommunikationsvorgang aus Senden eines Telegramms und Empfangen der Bestätigung abgeschlossen sein. Ist dies nicht der Fall, wird der Kommunikationsvorgang vom Master abgebrochen und im GUI der Kommunikationssoftware ein Hinweis auf eine Störung der Verbindung ausgegeben.

2.2.7 Auflistung der einzelnen Kommunikationsprotokolle

Das Identification-Byte (Byte 1) trägt bei allen Telegrammen den dezimalen Wert 85. Die entsprechenden Kommandos für Byte 3 können aus Tabelle 2.1 entnommen werden. Alle Werte in den Klammern entsprechen Dezimalwerten!

2.2.7.1 Master sendet Initialisierung

Byte 1:	IDENTIFICATION-Byte (= 85)
Byte 2:	Anzahl der Telegrammbytes inkl. Prüfsumme (= 9)
Byte 3:	Kommando (= 0)
Byte 4:	Anzahl der Datenbytes (= 2) ⇒ Byte 7 und Byte 8
Byte 5:	Datentyp der Initialisierungsdaten (= 0) ⇒ Integer
Byte 6:	Exponent n (= 4 bzw. 16) der Blockgröße 2^n
Byte 7:	Low-Byte der Blockgröße 2^n
Byte 8:	High-Byte der Blockgröße 2^n
Byte 9:	Prüfsumme

Die Initialisierung wird nach dem Start des Kommunikationsprogramms ausgeführt, um die Verbindung zwischen Computer (Master) und DSP (Slave) zu kontrollieren.
Weiters wird dem DSP in diesem Telegramm mitgeteilt, wie viele Variablen des RAM zu einem Block zusammengefasst werden sollen. Jedem Block wird dabei vom DSP ein Index zugeordnet. Anhand dieser Indizes kann der Inhalt des RAM's ausgelesen werden, wobei die Blöcke jeweils einzeln von einem Telegramm mit dem entsprechenden Index angefordert werden müssen.
Die Größe des RAM wird aufgrund der binären Datenverarbeitung stets über die Potenz 2^m bestimmt, wobei m eine vom Hersteller vorgegebene natürliche Zahl darstellt. Beispielsweise stellt das verwendete DSP-Board eine RAM-Größe von 2^{16} = 65536 Variablen zu jeweils 2 Byte zur Verfügung.
Wird die Blockgröße über die Potenz 2^n bestimmt, wobei n ebenfalls eine natürliche Zahl kleiner m repräsentiert, ist sichergestellt, dass das gesamte RAM in gleich große Bereiche unterteilt wird bzw. die Anzahl der Blöcke einer positiven ganzen Zahl entsprechen.

Abbildung 2.4 zeigt die schematische Darstellung der Aufteilung des RAM in einzelne Blöcke. Die RAM-Variablen sind stets vom Datentyp Integer und nehmen somit eine Breite von zwei Byte (16 Bit) ein.

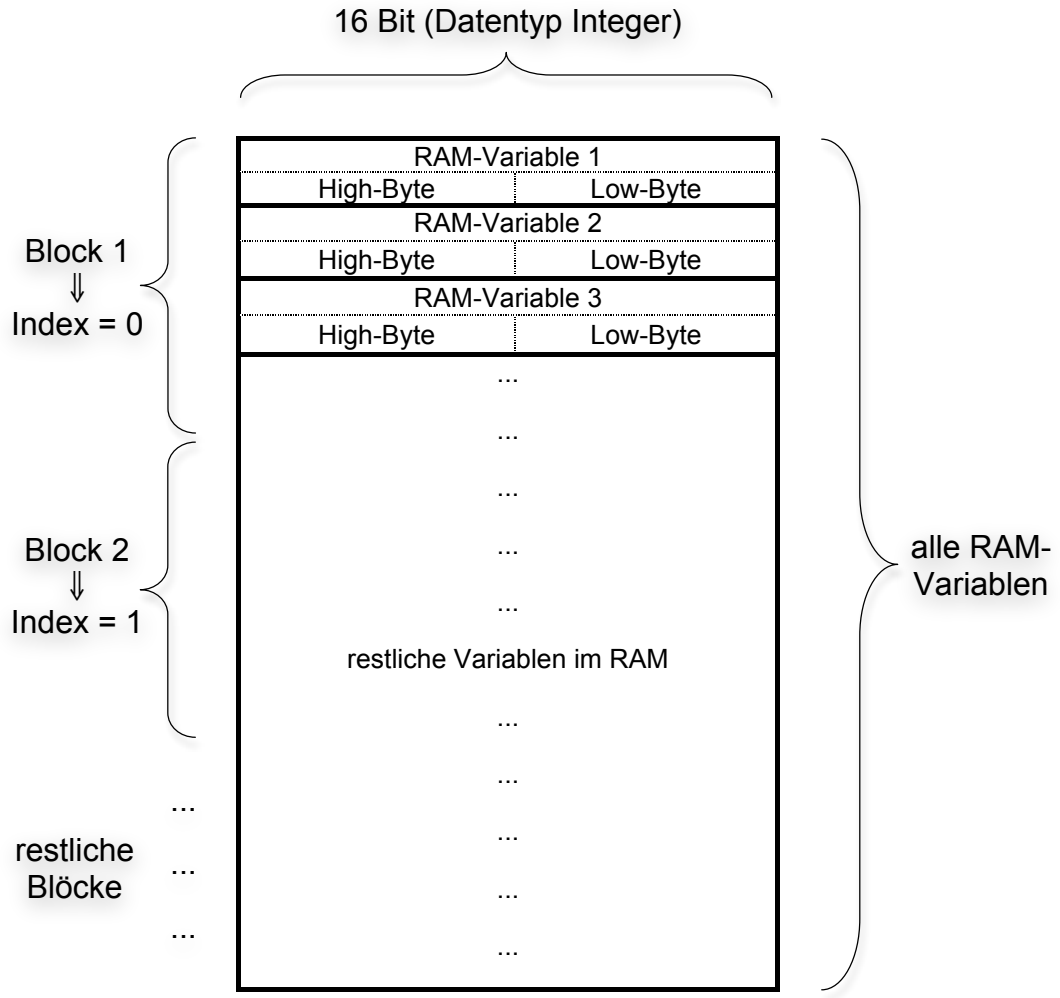


Abbildung 2.4: Aufteilung des RAM in gleich große Variablenblöcke
Die RAM-Variablen haben stets eine Breite von 16 Bit!

Je größer die Anzahl von Variablen in einem Block, desto weniger Telegramme müssen zum Auslesen aller RAM-Variablen versendet werden. Im Gegenzug dazu steigt die Wahrscheinlichkeit stark an, dass aufgrund der großen Anzahl an Bytes das Telegramm am Master fehlerhaft eintrifft. In diesem Fall müsste es erneut angefordert und die große Anzahl an enthaltenen Bytes geladen werden. Details zur Auswahl der Blockgröße sind in Abschnitt 2.3.6 dargestellt.

Byte 6 vermittelt dem DSP indirekt diese Anzahl an Variablen. Dieser betrachtet Byte 6 als Exponenten n und berechnet aus 2^n, wie viele Einheiten in einem Block zusammengefasst werden sollen.

$\Rightarrow$ Byte 6 = n

Mit Byte 7 und Byte 8 wird direkt die Größe 2^n, aufgeteilt in ein Low- und ein High-Byte (siehe Abschnitt 2.2.4), übertragen.

$\Rightarrow$ Byte 7 = 2^n (Low-Byte)

$\Rightarrow$ Byte 8 = 2^n (High-Byte)

Zur Darstellung der Blockgröße ist ein Integer-Datentyp ausreichend, wodurch sich für Byte 5 der Wert Null (siehe Abschnitt 2.2.3) ergibt.

2.2.7.2 DSP bestätigt Initialisierung

Byte 1:	IDENTIFICATION-Byte (= 85)
Byte 2:	Anzahl der Telegrammbytes inkl. Prüfsumme (= 5)
Byte 3:	Kommando (= 0)
Byte 4:	Anzahl der Datenbytes (= 0)
Byte 5:	Prüfsumme

2.2.7.3 Master sendet Adresse und Index

Byte 1:	IDENTIFICATION-Byte (= 85)
Byte 2:	Anzahl der Telegrammbytes inkl. Prüfsumme (= 11)
Byte 3:	Kommando (= 16)
Byte 4:	Anzahl der Adressbytes (= 4) $\Rightarrow$ Byte 7 bis Byte 10
Byte 5:	Datentyp der Adresse (= 1) $\Rightarrow$ Long
Byte 6:	Index der Variable (Wert zwischen 0 und 255)
Byte 7:	Adressbyte 1 (Low-Byte)
Byte 8:	Adressbyte 2
Byte 9:	Adressbyte 3
Byte 10:	Adressbyte 4 (High-Byte)
Byte 11:	Prüfsumme

Um im Bedarfsfall den gesamten Adressierungsbereich abdecken zu können, wird für die Adressen der Datentyp Long verwendet. Die Aufteilung der Adresse in einzelne übertragbare Bytes erfolgt nach dem gleichen Schema, wie unter Abschnitt 2.2.4 beschrieben.
Der Index zum Schreiben bzw. zum Lesen der betreffenden Variablen muss von Seiten des Masters vorgegeben und verwaltet werden (siehe Abschnitt 2.2.5).
Da die Zuweisung eines Index an eine Variable stets einzeln erfolgt, muss für jede Größe, deren Wert von Interesse ist, ein Telegramm an den DSP gesendet werden.

2.2.7.4 DSP bestätigt Empfang der Adresse und des Index

Byte 1:	IDENTIFICATION-Byte (= 85)
Byte 2:	Anzahl der Telegrammbytes inkl. Prüfsumme (= 7)
Byte 3:	Kommando (= 16)
Byte 4:	Anzahl der Datenbytes (= 2) ⇒ Byte 5 und Byte 6
Byte 5:	Datentyp der empfangenen Adresse (= 1) ⇒ Long
Byte 6:	Index der Variable (Wert zwischen 0 und 255)
Byte 7:	Prüfsumme

2.2.7.5 Master sendet Wert einer Variable (WRITE-Anforderung)

Byte 1:	IDENTIFICATION-Byte (= 85)	
Byte 2:	Anzahl der Telegrammbytes inkl. Prüfsumme (= 9 oder 11)	
Byte 3:	Kommando (= 32)	
Byte 4:	Anzahl der Datenbytes (= 2 oder 4) ⇒ Byte 7 bis vorletztes Byte	
Byte 5:	Datentyp der Variable (= 0 oder 1) ⇒ Integer oder Long	
Byte 6:	Index der betreffenden Variable (Wert zwischen 0 und 255)	
Byte 7:	Datenbyte 1 (Low-Byte)	
Byte 8:	Datenbyte 2	
(Byte 9:	Datenbyte 3)	} zusätzlich bei Datentyp Long
(Byte 10:	Datenbyte 4)	
letztes Byte:	Prüfsumme	

Die Gesamtanzahl an Telegrammbytes hängt vom Datentyp der jeweiligen Variable ab (siehe Abschnitt 2.2.3). Die Aufteilung der Variable in einzelne Datenbytes erfolgt nach dem in Abschnitt 2.2.4 erläutertem Prinzip. Werte von Variablen werden vom Master stets einzeln an den DSP gesendet.

2.2.7.6 DSP bestätigt Empfang des Variablenwerts

Byte 1:	IDENTIFICATION-Byte (= 85)
Byte 2:	Anzahl der Telegrammbytes inkl. Prüfsumme (= 7)
Byte 3:	Kommando (= 32)
Byte 4:	Anzahl der Datenbytes (= 2) ⇒ Byte 5 und Byte 6
Byte 5:	Datentyp der empf. Variable (= 0 oder 1) ⇒ Integer oder Long
Byte 6:	Index der Variable (Wert zwischen 0 und 255)
Byte 7:	Prüfsumme

2.2.7.7 Master fordert vom DSP bis zu acht Variablenwerte

Mit diesem Telegramm können, je nach Bedarf, bis zu acht Variablenwerte vom DSP angefordert werden. Jede gewünschte Variable wird anhand eines Index, der bereits vorher einer Variablenadresse zugewiesen worden ist, ausgelesen (siehe Abschnitt 2.2.5). Jeder Index wird dabei als einzelnes Byte ab Position sechs im Telegramm eingefügt.

Sind mehr als acht Werte für den Master von Interesse, muss ein weiters Telegramm mit den betreffenden Indizes erstellt und an den DSP gesendet werden.
Byte 5 enthält die Information, welchen Datentypen die angeforderten Variablen entsprechen. Das niedrigstwertige Bit bzw. das „Least Significant Bit“ (LSB) von Byte 5 steht dabei stellvertretend für den Typ der ersten angeforderten Variable, welche durch Index 1 in Byte 6 repräsentiert wird. Das zweite Bit in Byte 5 dient der Typen-Information über Variable 2 (Index 2 in Byte 7) usw.

Byte 1:	IDENTIFICATION-Byte (= 85)
Byte 2:	Anzahl der Telegrammbytes inkl. Prüfsumme (= 7 bis 14)
Byte 3:	Kommando (= 48)
Byte 4:	Anzahl der Indizes/Variablen (= 1 bis 8) ⇒ 6. bis vorletztes Byte
Byte 5:	Datentypen der Variablen
Byte 6:	Index der 1. angeforderten Variable (Wert zwischen 0 und 255)
...	... } bis zu 7 weitere Indizes möglich
...	...
...	...
letztes Byte:	Prüfsumme

2.2.7.8 DSP sendet angeforderte Anzahl an Variablenwerten

Byte 1:	IDENTIFICATION-Byte (= 85)
Byte 2:	Anzahl der Telegrammbytes inkl. Prüfsumme
Byte 3:	Kommando (= 48)
Byte 4:	Anzahl der Datenbytes ⇒ Byte 6 bis vorletztes Byte
Byte 5:	Datentypen der folgenden Variablen
Byte 6:	Index der 1. angeforderten Variable (Wert zwischen 0 und 255)
Byte 7:	Datenbyte 1 der 1. Variable (Low-Byte)
Byte 8:	Datenbyte 2 der 1. Variable
(Byte 9:	Datenbyte 3 der 1. Variable) } zusätzlich bei Datentyp Long
(Byte 10:	Datenbyte 4 der 1. Variable)
...	Index 2 } bis zu 7 weitere Variablen möglich
...	2 bis 4 Datenbytes
...	Index 3
...	2 bis 4 Datenbytes
...	...
...	...
letztes Byte:	Prüfsumme

Mit diesem Telegramm werden vom DSP bis zu acht Variablen als Antwort an den Master retour gesendet. In der Reihenfolge der Datenbytes (ab Byte 6) steht immer zuerst die Indexnummer der Variable und im Anschluss daran, je nach Datentyp, die zwei bis vier Datenbytes der betreffenden Variable. Diese Reihenfolge, zuerst Index danach Datenbytes, wiederholt sich für alle gewünschten Werte in der Nachricht.
Details zur Verarbeitung der einzelnen Datenbytes finden sich unter Abschnitt 2.2.4.

2.2.7.9 Master sendet Softauge-Adresse und Kanalnummer (Index)

Byte 1:	IDENTIFICATION-Byte (= 85)
Byte 2:	Anzahl der Telegrammbytes inkl. Prüfsumme (= 11)
Byte 3:	Kommando (= 64)
Byte 4:	Anzahl der Adressbytes (= 4) ⇒ Byte 7 bis Byte 10
Byte 5:	Datentyp der Adresse (= 1) ⇒ Long
Byte 6:	Kanalnummer für Variable (Wert zwischen 1 und 4)
Byte 7:	Adressbyte 1 (Low-Byte)
Byte 8:	Adressbyte 2
Byte 9:	Adressbyte 3
Byte 10:	Adressbyte 4 (High-Byte)
Byte 11:	Prüfsumme

Das DSP-Board ist in der Lage, vier analoge Kanäle zum Anschluss eines Oszilloskops zu betreiben. Diese Funktion wird als „Softauge“ bezeichnet.
Welche Variable am jeweiligen Kanal ausgeben werden soll, wird dem DSP anhand dieses Telegramms mitgeteilt. Es enthält die Adresse der auszugebenden Variable und die gewünschte Kanalnummer.
Da Adressen vom Datentyp Long sind, sind zur Darstellung im Telegramm vier Bytes notwendig. Die Aufteilung der Adresse in die einzelnen Telegrammbytes wird unter Abschnitt 2.2.4 erläutert.
Die Zuweisung einer Variable zu einem „Softauge-Kanal“ erfolgt stets einzeln in einem Telegramm.

2.2.7.10 DSP bestätigt Empfang der Softauge-Adresse und der Kanalnummer

Byte 1:	IDENTIFICATION-Byte (= 85)
Byte 2:	Anzahl der Telegrammbytes inkl. Prüfsumme (= 7)
Byte 3:	Kommando (= 64)
Byte 4:	Anzahl der Datenbytes (= 2) ⇒ Byte 5 und Byte 6
Byte 5:	Datentyp der empfangenen Adresse (= 1) ⇒ Long
Byte 6:	Kanalnummer der Variable (Wert zwischen 1 und 4)
Byte 7:	Prüfsumme

2.2.7.11 Master sendet Zoomfaktor eines Softauge-Kanals

Byte 1:	IDENTIFICATION-Byte (= 85)
Byte 2:	Anzahl der Telegrammbytes inkl. Prüfsumme (= 9 oder 11)
Byte 3:	Kommando (= 80)
Byte 4:	Anzahl der Datenbytes (= 2 oder 4) ⇒ Byte 7 bis vorletztes Byte
Byte 5:	Datentyp des Zoomwertes (= 0 oder 1) ⇒ Integer oder Long
Byte 6:	Kanalnummer (Wert zwischen 1 und 4)
Byte 7:	Datenbyte 1 (Low-Byte)
Byte 8:	Datenbyte 2
(Byte 9:	Datenbyte 3) } zusätzlich bei Datentyp Long
(Byte 10:	Datenbyte 4) } zusätzlich bei Datentyp Long
letztes Byte:	Prüfsumme

Jedem der vier „Softauge-Kanäle“ kann ein Faktor als Zoomwert zugewiesen werden, um die dem Kanal entsprechende Variable am Oszilloskop passend darstellen zu können.
Die gewünschten Zoomwerte werden, so wie die Adressen der Variablen, stets einzeln an den DSP übertragen.
Je nach Größe des Faktors, entspricht dieser dem Datentyp Long oder Integer. Details zur Zuweisung des passenden Datentyps finden sich im Abschnitt 2.2.3.

2.2.7.12 DSP bestätigt Empfang des Zoomfaktors eines Softauge-Kanals

Byte 1:	IDENTIFICATION-Byte (= 85)
Byte 2:	Anzahl der Telegrammbytes inkl. Prüfsumme (= 7)
Byte 3:	Kommando (= 80)
Byte 4:	Anzahl der Datenbytes (= 2) ⇒ Byte 5 und Byte 6
Byte 5:	Datentyp des empfangenen Zoomwertes (= 0 oder 1)
Byte 6:	Kanalnummer (Wert zwischen 1 und 4)
Byte 7:	Prüfsumme

2.2.7.13 Master fordert Variablenblock des RAM auszulesen

Byte 1:	IDENTIFICATION-Byte (= 85)
Byte 2:	Anzahl der Telegrammbytes inkl. Prüfsumme (= 7)
Byte 3:	Kommando (= 96)
Byte 4:	Anzahl der Datenbytes (= 2) ⇒ Byte 5 und Byte 6
Byte 5:	Low-Byte des Index vom gewünschten Datenpaket
Byte 6:	High-Byte des Index vom gewünschten Datenpaket
Byte 7:	Prüfsumme

Wie in Abbildung 2.4 dargestellt, wird bei der Initialisierung das RAM des DSP in mehrere Blöcke bzw. Gruppen aufgeteilt. Diesen Blöcken wird dabei vom DSP jeweils ein Index zugewiesen, mit dessen Hilfe die Variablen ausgelesen werden können. Dabei wird der ersten Gruppe der Index 0 zugeschrieben, der zweiten Gruppe der Index 1 usw.
Einzelne Variablen können nicht angefordert werden, nur eine gesamte Gruppe, in der sich die gewünschte Variable befindet!
Jeder gewünschte Variablenblock muss mit einem eigenen Telegramm ausgelesen werden. Wird zum Beispiel ein 65536 Variablen umfassendes RAM (dies entspricht der RAM-Größe des verwendeten DSP-Boards) in Blöcke zu je 16 Variablen aufgeteilt, entstehen 4096 Gruppen. Um nun alle Variablen des RAM auszulesen, sind dementsprechend 4096 Telegramme notwendig.
Aus diesem Grund ist es notwendig, für die Übertragung des Index zwei Bytes vorzusehen. Mit einem Byte (8 Bit) wäre es möglich, Indizes nur von 0 bis 255 zu unterscheiden (2^8 Werte). Mit 16 Bit wächst diese Möglichkeit auf einen Bereich von 0 bis 65535 (2^{16} Werte) an. Details zur Aufteilung der Adresse in übertragbare Bytes finden sich in Abschnitt 2.2.4.

Vorteil dieser Aufteilung ist, dass man auf unterschiedliche Bereiche des RAM zugreifen kann. Wird beispielsweise das Auslesen des gesamten RAM kurz vor Abschluss wegen einer kurzfristigen Störung der Verbindung unterbrochen, kann nach Beseitigung der Störung das Auslesen an der unterbrochenen Stelle fortgesetzt werden.

2.2.7.14 DSP sendet angeforderten Variablenblock des RAM

Byte 1:	IDENTIFICATION-Byte (= 85)
Byte 2:	Anzahl der Telegrammbytes inkl. Prüfsumme
Byte 3:	Kommando (= 96)
Byte 4:	Anzahl der Datenbytes ⇒ Byte 7 bis vorletztes Byte
Byte 5:	Low-Byte des Index vom gewünschten Datenpaket
Byte 6:	High-Byte des Index vom gewünschten Datenpaket
Byte 7:	Low-Byte der 1. Variable im Block
Byte 8:	High-Byte der 1. Variable im Block
Byte 9:	Low-Byte der 2. Variable im Block
Byte 10:	High-Byte der 2. Variable im Block
Byte 11:	Low-Byte der 3. Variable im Block
Byte 12:	High-Byte der 3. Variable im Block
...	...
...	...
Letztes Byte:	Prüfsumme

Die Anzahl der im Telegramm enthaltenen Variablen hängt von der Größe des Variablenblocks ab.
Alle RAM-Variablen sind vom Typ Integer. Aus diesem Grund müssen sie vom Absender zur Übertragung in zwei Bytes aufgeteilt und vom Empfänger wieder zusammengefügt werden.
Näheres zur Verarbeitung von Datenbytes findet sich unter Abschnitt 2.2.4.

2.2.7.15 Master fordert Variablenblock des EEPROM auszulesen

Byte 1:	IDENTIFICATION-Byte (= 85)
Byte 2:	Anzahl der Telegrammbytes inkl. Prüfsumme (= 7)
Byte 3:	Kommando (= 112)
Byte 4:	Anzahl der Datenbytes (= 2) ⇒ Byte 5 und Byte 6
Byte 5:	Low-Byte des Index vom gewünschten Datenpaket
Byte 6:	High-Byte des Index vom gewünschten Datenpaket
Byte 7:	Prüfsumme

Das Auslesen des EEPROM-Inhalts erfolgt nach dem selben Schema wie das Auslesen des RAM's. Die Telegramme unterscheiden sich nur in deren Kommando.
Die Blockgrößen des EEPROM umfassen jedoch stets 16 Variablen und sind nicht veränderbar.

2.2.7.16 DSP sendet angeforderten Variablenblock des EEPROM

Byte 1:	IDENTIFICATION-Byte (= 85)
Byte 2:	Anzahl der Telegrammbytes inkl. Prüfsumme
Byte 3:	Kommando (= 112)
Byte 4:	Anzahl der Datenbytes ⇒ Byte 7 bis vorletztes Byte
Byte 5:	Low-Byte des Index vom gewünschten Datenpaket
Byte 6:	High-Byte des Index vom gewünschten Datenpaket
Byte 7:	Low-Byte der 1. Variable im Block
Byte 8:	High-Byte der 1. Variable im Block
Byte 9:	Low-Byte der 2. Variable im Block
Byte 10:	High-Byte der 2. Variable im Block
Byte 11:	Low-Byte der 3. Variable im Block
Byte 12:	High-Byte der 3. Variable im Block
...	...
...	...
Letztes Byte:	Prüfsumme

Die weitere Verarbeitung der Datenbytes aus dem Telegramm ist ferner mit der Verarbeitung der RAM-Variablen vergleichbar.

2.3 Parametrierungssoftware

2.3.1 Allgemeines und Voraussetzungen

Um elektrische Maschinen an deren unterschiedliche Anwendungen anpassen zu können, müssen die zugehörigen Umrichter entsprechend parametriert werden.
Die mit MATLAB erstellte und hier beschriebene grafische Benutzeroberfläche (GUI) bietet die Möglichkeit, diesen Parametrierungsvorgang übersichtlich durchzuführen.
Zu diesem Zweck ist es notwendig, den Umrichter mit einem DSP-Board auszustatten. Um neben einer seriellen Verbindung eine TCP/IP-Verbindung zu ermöglichen, ist ein zusätzlicher SCI-Ethernet-Adapter notwendig. Der Adapter dient dabei als Webserver, der über eine ihm fix zugewiesene IP-Adresse via TCP/IP-Protokoll erreicht werden kann.
Durch die Verwendung des Webservers kann die Verbindung zwischen Computer und DSP-Board über eine Netzwerk- bzw. über eine Internetanbindung erfolgen. Falls erforderlich, ist es jedoch auch möglich, die Verbindung über eine direkte Ethernetverkabelung herzustellen.
Die Art der Verbindung, TCP/IP oder seriell, ist für das unter Punkt 2.2 erläuterte Protokoll nicht von Bedeutung. Man muss bei der Initialisierung des DSP nur die entsprechenden Verbindungsdaten einstellen. Für TCP/IP ist eine Angabe der IP-Adresse des Webservers und den zu verwendenden Port notwendig. Die serielle Verbindung erfordert die Bekanntgabe des korrekten Com-Ports und der Baudrate. Sind die Daten korrekt, öffnet MATLAB eine entsprechende Verbindung, die für die Übertragung der oben erläuterten Telegramme genutzt wird.
Um das MATLAB-GUI am Computer benutzen zu können, ist eine MATLAB-Version R2009a oder höher notwendig. Handelt es sich bei dem GUI um eine ausführbare Datei, kann anstelle der vollständigen MATLAB-Version die frei erhältliche MATLAB-Runtime (MCR) in entsprechender Version installiert werden.
Ein wesentlicher Bestandteil für die Kommunikation zwischen DSP und Computer stellt das Map-File dar. Dieses wird beim Kompilieren der DSP-Firmware automatisch erzeugt. Es enthält eine Auflistung aller Variablen bzw. Parameter der DSP-Firmware sowie deren zugehörige Adressen. Anhand dieses Map-Files, dessen Pfad beim Projektstart im GUI hinterlegt werden muss, können die Parameter ausgelesen oder geändert werden. Es ist somit von wesentlicher Bedeutung, dass stets mit einem aktuellen Map-File gearbeitet wird!

Um zu gewährleisten, dass das Map-File von der MATLAB-Software korrekt interpretiert werden kann, muss bei dessen Erstellung auf fünf Vorgaben geachtet werden:

- Der Beginn der Auflistung muss mit der Überschrift „address name“ beginnen.
- In der Zeile nach der Überschrift ist eine Trennzeile mit Bindestrichen einzufügen („------- ----“).
- Der Abstand zwischen der jeweiligen Adresse und dem Namen muss vier Leerzeichen (Spaces) betragen.
- Das Ende der Liste ist mit „ [“ abzuschließen.
- Jeder Variablenname muss mit einem zusätzlichen Zeichen beginnen (z.B. ' _ '), da dies im MATLAB-GUI entfernt wird.

Als Beispiel für diese Syntax dient Abbildung 2.5:

```
beliebiger Text

address    name
--------   ----
00000400   __STACK_SIZE
00000d00   _PieVectTable
00008000   _FLASH_ini
0000904d   _Led1
0000904a   _Led2

[beliebiger Text]
```

Abbildung 2.5: Beispiel für Syntax des Map-Files

Weiters ist zu beachten, dass dieses beim Kompilieren erzeugte Map-File in einem Ordner mit dem Namen „Debug“ abgelegt wird. Dieser Ordner wird ebenfalls während des Flashvorgangs erzeugt und an einem vorgegebenen Ort gespeichert. Aktualisiert man nun die Firmware des DSP, so wird nicht nur das Map-File automatisch neu generiert, sonder der gesamte „Debug-Ordner“. Alle bisherigen Dateien im Ordner „Debug“ werden somit durch einen Kompiliervorgang gelöscht. Aus diesem Grund ist darauf zu achten, dass sich der Projektpfad des MATLAB-GUI, oder auch andere Dateien, nicht auf diesen Ordner beziehen!

2.3.2 Start und Speichern eines Projekts

Zum Start eines Projekts muss die Funktion IEAM_Com.m bzw. IEAM_Com.p ausgeführt werden. Dies geschieht, indem „IEAM_Com“ in das „Command Window“ von Matlab eingetragen und durch die Eingabe von „Enter“ bestätigt wird. Dabei ist darauf zu achten, dass als „Current Directory“ der spezifische Softwareordner des MATLAB-GUI gewählt wurde!
Durch den Start der Funktion werden zwei Fenster geöffnet. Das Fenster im Hintergrund stellt die Oberfläche des GUI ohne Variablen dar. Das Fenster im Vordergrund fordert dazu auf, das Programm zu beenden oder ein neues Projekt anzulegen bzw. ein existierendes Projekt zu laden. Abbildung 2.6 zeigt die beiden Fenster und die Auswahlmöglichkeiten.
Wird die Option zum Start bzw. zum Laden eines Projektes ausgewählt und mit „OK“ bestätigt, öffnet sich ein Fenster zur Auswahl des bereits unter Punkt 2.3.1 erläuterten Map-Files. Bei Auswahl eines korrekten MAP-Files speichert das GUI den Pfad zu dieser Datei und liest die enthaltene Auflistung der Variablen inklusive Adressen ein. Handelt es sich bei der ausgewählten Datei um keine „*.map-Datei“ oder sind keine verwertbaren Variablen in der Auflistung vorhanden, wird eine entsprechende Fehlermeldung ausgegeben (siehe Abbildung 2.7a und b).

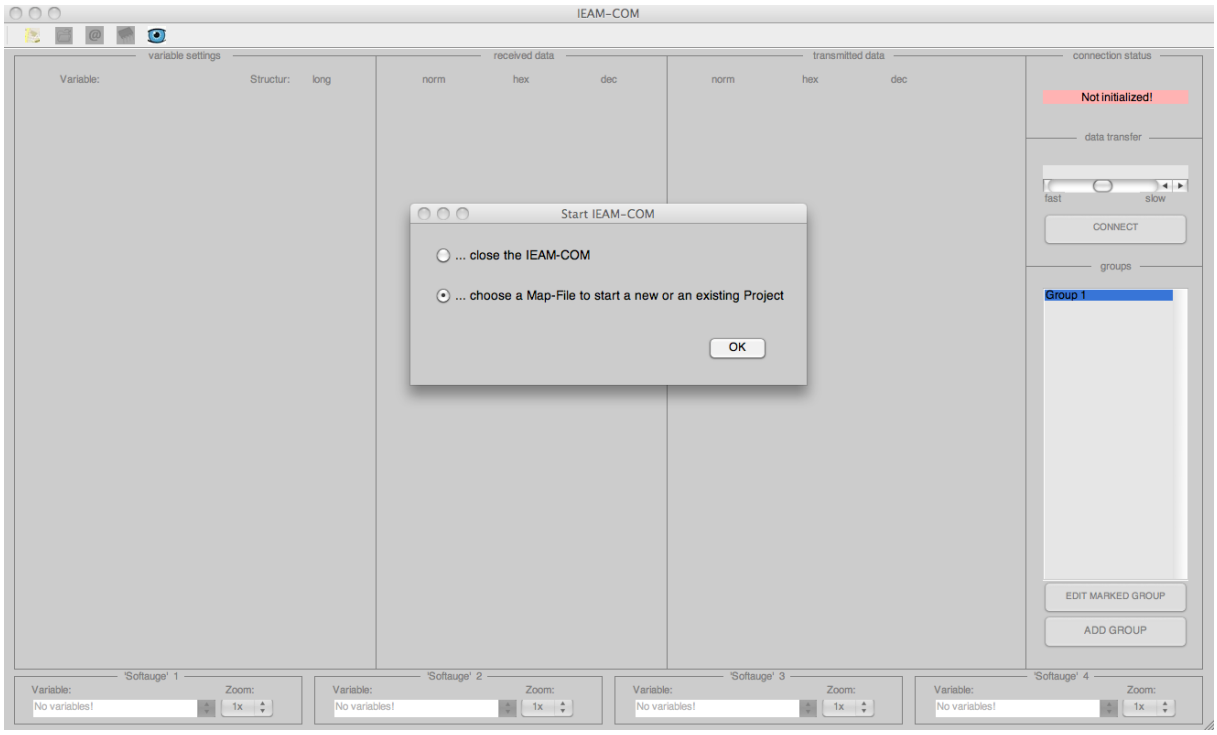
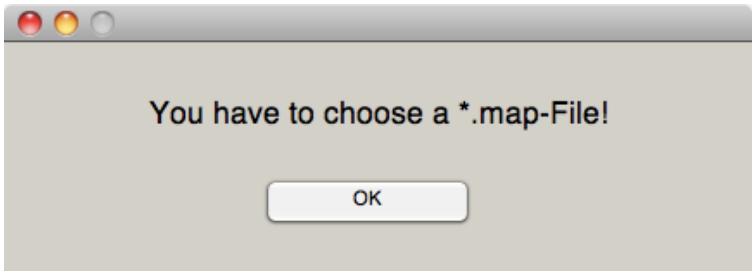


Abbildung 2.6[1]: MATLAB-GUI nach Start der Funktion IEAM_Com

a)



b)

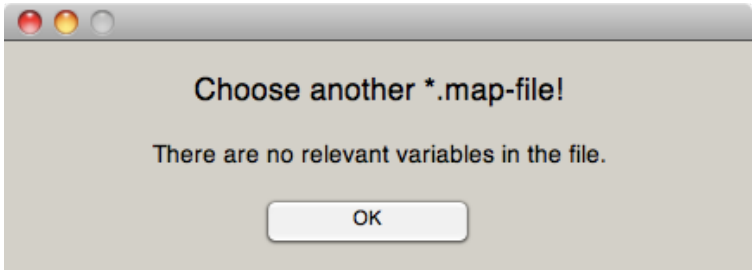


Abbildung 2.7[1]: Fehlermeldungen bei Auswahl eines falschen Map-Files

a) Die gewählte Datei ist keine „.map-Datei“.*

b) In der gewählten Datei sind keine Variablen aufgelistet, bzw. die Variablen können aufgrund einer falschen Syntax nicht gefunden werden.

[1] Quelle: MATLAB-GUI

Neben der Variablen- und Adressauflistung in dieser Datei, die für die Kommunikation des GUI notwendig sind, stellen Name und Pfad des Map-Files gegebenenfalls die Basis für die zu erzeugende oder zu ladende Projektdatei dar. In der Projektdatei werden alle relevanten Daten eines Projekts gesichert, um beim nächsten Start wieder verfügbar zu sein.
Der aktuelle Projektname wird vom GUI automatisch vom Dateinamen des ausgewählten Map-Files übernommen. Zudem wird ein Pfad erstellt, der den Speicherort für die entsprechende Projektdatei beschreibt. Dieser Projektpfad gleicht dem Pfad des Map-Files, außer dieses befindet sich in einem Ordner namens „Debug“. In diesem Fall wird ein Pfad generiert, der auf einen Speicherort direkt außerhalb dieses Ordners verweist. Das ist notwendig, da bei einem neuerlichen Kompiliervorgang Dateien innerhalb von Debug, wie in Abschnitt 2.3.1 beschrieben, gelöscht werden.
Wird es vom Benutzer des Programms bevorzugt, selbst einen Pfad und einen Namen für eine neue Projektdatei auszuwählen, kann dies vor dem Start des MATLAB-GUI in den Software-Parametern eingestellt werden (siehe Tabelle 2.4, Variable „auto_projectload“). Aber auch bei der manuellen Vergabe des Speicherortes für die Projektdatei ist darauf zu achten, den „Debug-Ordner“ des Map-Files zu meiden! Weiters muss es sich bei der erstellten Projektdatei um eine „*.mat-Datei“ handeln. Ansonsten wird eine Fehlermeldung angezeigt und eine erneute Eingabe des Projektnamens gefordert (siehe Abbildung 2.8). Für nähere Details zu den Software-Parametern sei auf Punkt 2.3.8 verwiesen.

Abbildung 2.8[1]: Meldung bei Auswahl eines falschen Projekt-Datentyps

Befindet sich an dem vom aktuellen Projektpfad beschrieben Speicherort bereits eine Projektdatei und gleicht deren Name dem des aktuellen Projekts, so werden die Daten aus dieser Datei automatisch in das GUI geladen! Dies gilt sowohl für die manuelle als auch für die automatische Erstellung eines Projekts.
Beim Beenden des GUI bzw. beim Auswählen eines anderen Projekts werden alle Variablen und Einstellungen selbständig in der aktuellen Projektdatei gesichert und als „*.mat-Datei“ entsprechend dem erzeugten Projektpfad abgelegt. Zudem wird die „ini-Datei“ des MATLAB-GUI erzeugt bzw. aktualisiert und im Verzeichnis der MATLAB-Funktionen abgelegt. Diese Datei beinhaltet Informationen über das zuletzt verwendete Projekt und dessen Speicherort. Beim nächsten Start des GUI wird mit den Angaben aus der ini-Datei, im entsprechenden Auswahlfenster, das zuletzt

[1] Quelle: MATLAB-GUI

verwendete Map-File markiert. Dies bringt den Vorteil, dass bei Bedarf das letzte Projekt schneller ausgewählt werden kann.
Ist die Projektdatei im Zuge des Projektstart ausgewählt worden, wird, falls noch nicht vorhanden, an dem generierten Speicherort automatisch ein Ordner mit dem Projektnamen und dem Zusatz „LOG“ erzeugt. Dieser dient dazu, gegebenenfalls Daten von „The Eye“ zu sichern. Die Funktion „The Eye“ wird unter Punkt 2.3.7 beschrieben.

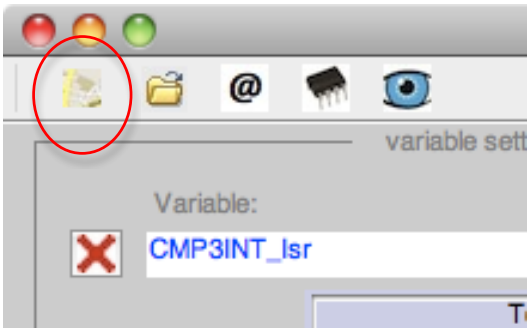


Abbildung 2.9[1]: Button zum Laden eines neuen Projekts

Will man ein neues Projekt starten, ohne das MATLAB-GUI zu beenden, muss der in Abbildung 2.9 markierte Schaltfläche in der Menüleiste des GUI betätigt werden. Alle bisherigen Variablen und Einstellungen werden in der bis dahin aktuellen Projektdatei gesichert und der Vorgang zur Auswahl eines Map-Files beginnt nach dem selben Schema wie beim Start des MATLAB-GUI.

2.3.3 Variablenauswahl

Wie bereits unter Punkt 2.3.2 erläutert, wird beim Start des MATLAB-GUI die entsprechende Projektdatei erstellt bzw., wenn bereits vorhanden, geladen.
Handelt es sich um ein neues Projekt oder sind in dem bestehenden und geladenen Projekt keine Variablen vorhanden, wird ein Fenster zur Auswahl der GUI-Variablen geöffnet. Dieses Variablenfenster ist in Abbildung 2.10 dargestellt.
Die linke Spalte beinhaltet alle Variablen, die im ausgewählten Map-File aufgelistet sind. Es können mehrere Variablen gleichzeitig markiert werden und mit dem Button „Add =>“ in die rechte Spalte übertragen werden. Sind bereits 24 Elemente in der rechten Spalte eingetragen, wird diese Schaltfläche deaktiviert.
Um eine gewünschte Variable anhand seiner Anfangsbuchstaben im Map-File suchen zu können, muss die linke Spalte mittels eines einfachen Mausklicks markiert werden. Durch die Eingabe eines Suchstrings, dieser wird in der obersten Zeile blau dargestellt, wird die Liste in der linken Spalte auf all die Variablen eingeschränkt, deren Name mit diesem String beginnen. Mit der Escape-Taste oder durch das Löschen des Strings wird die Einschränkung der Variablenliste wieder aufgehoben.

[1] Quelle: Ausschnitt aus dem MATLAB-GUI

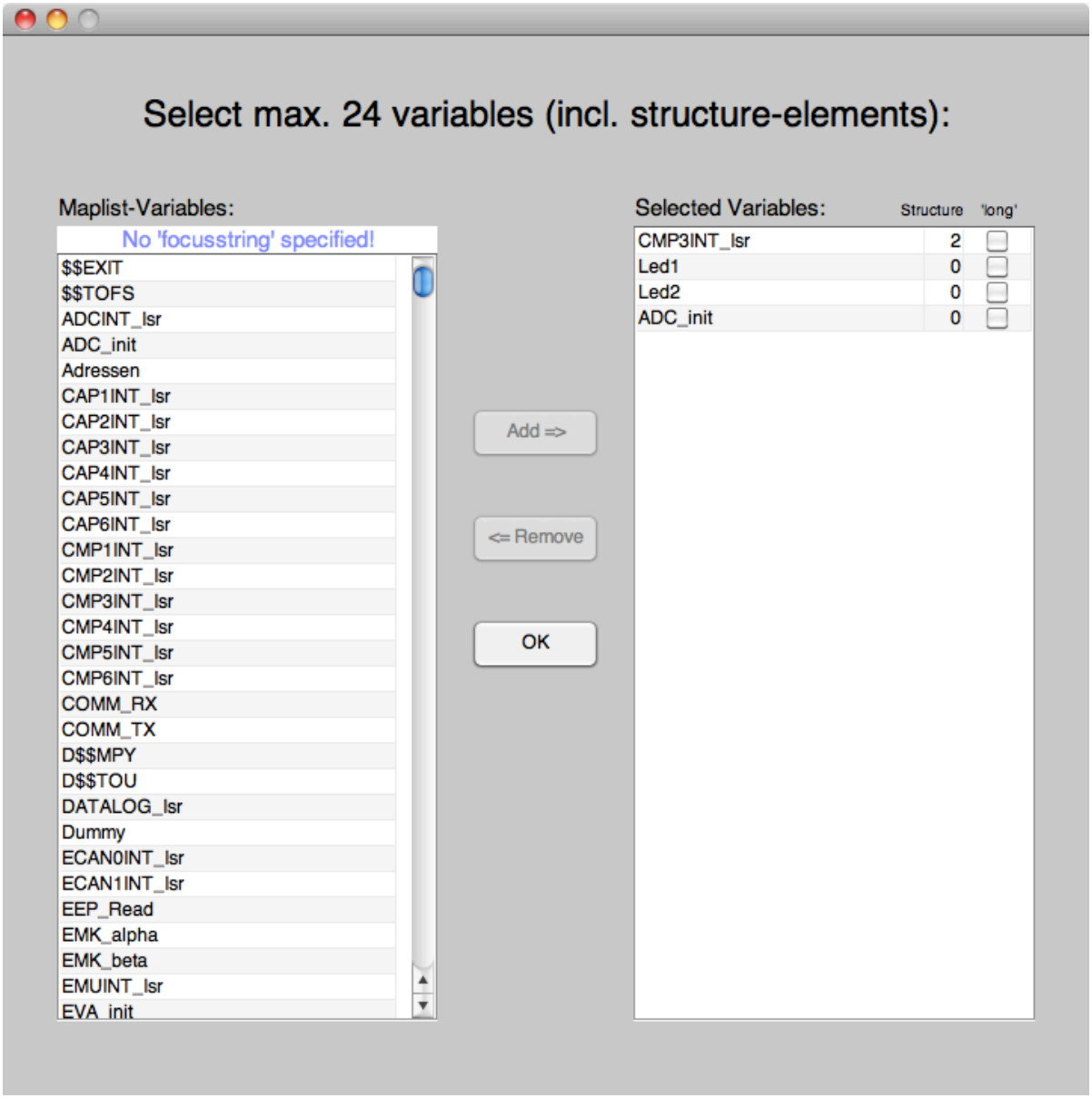


Abbildung 2.10[1]*: Variablenfenster zur Verwaltung der Variablen im GUI*

Die rechte Spalte beinhaltet alle Variablen, die bereits ausgewählt wurden bzw. die gegebenenfalls bereits im GUI vorhandenen sind. Es können auch hier mehrere Variablen gleichzeitig markiert und über den Button „<= Remove“ aus der Liste entfernt werden. Wird eine Variable, die bereits im GUI vorhanden ist, aus dieser Spalte entfernt, so wird diese mit der Betätigung des „OK-Buttons“ auch aus dem GUI entfernt.
Weiters besteht die Möglichkeit, in dieser Spalte anhand der Checkbox den Datentyp der jeweiligen Variable vorzugeben. Ist die Checkbox aktiviert, handelt es sich bei der Variable um den Typ Long. Andernfalls ist diese vom Typ Integer.
Ein Zahlenwert größer Null unter der Rubrik „Structure“ signalisiert, dass diese Variable im GUI als Strukturvariable mit der entsprechenden Anzahl an Struktur-

[1] Quelle: MATLAB-GUI

elementen verwendet wird. Details über die Verwendung und die Bedeutung von Strukturvariablen finden sich unter Punkt 2.3.5.
Beim Hinzufügen von Variablen aus der linken in die rechte Spalte muss berücksichtigt werden, dass auch die erwähnten Strukturelemente zur Gesamtanzahl der Variablen beitragen. Ist zum Beispiel eine Variable mit 2 Strukturelementen im GUI vorhanden, können nur noch maximal 21 Variablen eingefügt werden.
Um das Fenster zur Auswahl und zur Verwaltung der GUI-Variablen unabhängig vom Projektstart öffnen zu können, muss der in Abbildung 2.11 markierte Knopf in der Menüleiste des GUI betätigt werden.

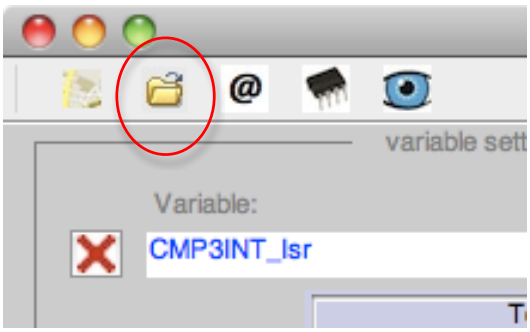


Abbildung 2.11[1]: Button zum Öffnen des Variablenfensters

Durch die Betätigung des „OK-Buttons“ werden die in der rechten Spalte aufgelisteten Variablen in das MATLAB-GUI übertragen und die zugehörigen Adressen der Variablenverwaltung im Hintergrund des GUI zugeführt.

2.3.4 Kommunikationseinstellungen

Das Fenster zur Einstellung der Kommunikationsparameter dient der Initialisierung bzw. der Überprüfung der Verbindung zwischen dem MATLAB-GUI und dem DSP-Board. Es erscheint, sofern noch keine Initialisierung erfolgte, stets nach der Auswahl des „OK-Buttons“ im Variablenfenster oder nach dem Betätigen des in Abbildung 2.12 markierten Knopfes in der Menüleiste des GUI.

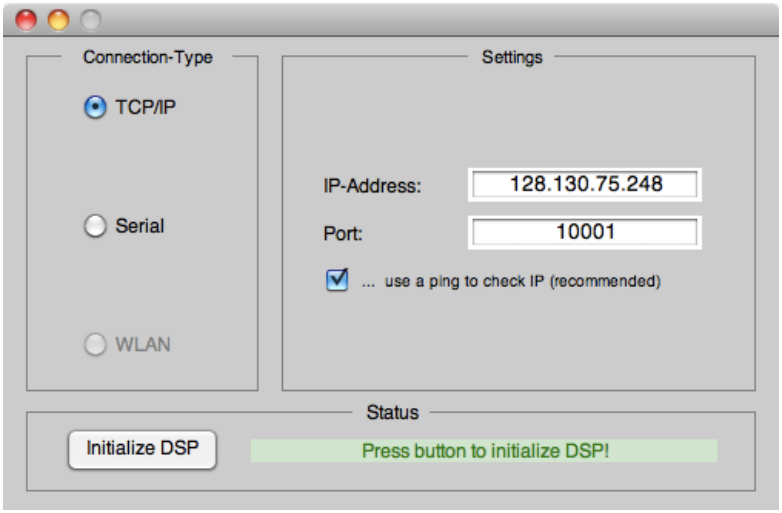
Abbildung 2.12[1]: Button zum Öffnen des Verbindungsparamter-Fenster

[1] Quelle: Ausschnitt aus dem MATLAB-GUI

Je nach ausgewähltem Verbindungstyp unterscheidet sich das Fenster, siehe Abbildung 2.13, entsprechend der einzustellenden Parameter.
Mit der im Moment verfügbaren Hardware kann als Verbindungstyp „TCP/IP“ oder „Seriell“ ausgewählt werden. Weitere Typen werden voraussichtlich mit der nächsten Generation des DSP-Boards möglich sein. Der Typ „WLAN“ dient im Moment daher nur als Platzhalter für Erweiterungen.
Die Verwendung der TCP/IP-Verbindung hat den Vorteil, dass Computer und DSP-Board räumlich getrennt voneinander verwendet werden können, sofern beide über eine Internetverbindung verfügen. Es ist jedoch auch eine direkte Verbindung via Ethernetkabel möglich.
Zum Aufbau dieser Verbindung ist es notwendig, in dem dafür vorgesehenen Bereich die IP-Adresse des DSP-Board-Webservers im IPv4-Format (Beispiel siehe Abbildung 2.13a) einzutragen. Weiters wird die Nummer des Ports benötigt, über den der Datenverkehr abgewickelt werden kann. Beide Daten sind vom Webserver vorgegeben und müssen für einen Verbindungsaufbau bekannt sein.
Die Initialisierung des DSP wird mit der Betätigung des Buttons „Initialize DSP“ gestartet. Während dieses Vorgangs kann das Fenster nicht geschlossen werden.
Der Text rechts vom Initialisierungsknopf informiert über den aktuellen Status des Verbindungsaufbaus. Ist die Verbindung zum DSP hergestellt, schließt das Fenster selbstständig. Andernfalls weist der Statustext auf einen Fehler hin und das Fenster bleibt bestehen.
Wird die Initialisierung mit einer nicht existierenden IP-Adresse gestartet, versucht MATLAB eine entsprechende TCP/IP-Verbindung herzustellen, bis dieser Vorgang durch ein Timeout abgebrochen wird. Dieses Timeout ist von MATLAB vorgegeben und liegt bei etwa zwei Minuten. Erst danach wird die entsprechende Statusmeldung angezeigt und kann gegebenenfalls das Fenster geschlossen werden.
Um diesen Vorgang im Fehlerfall abzukürzen, wird vom GUI beim Initialisierungsstart die Existenz der IP-Adresse mit Hilfe eines „Pings“ überprüft. Durch diesen einfachen Test verkürzt sich im Fehlerfall die Zeit bis zur diesbezüglichen Statusmeldung auf 20 bis 30 Sekunden.
Falls das DSP-Board jedoch eine Internetanbindung hat, deren Sicherheitseinstellung das Antworten auf „Pings“ verhindert, wird vom GUI trotz korrekter IP-Adresse keine Verbindung aufgebaut. Der Grund dafür liegt darin, dass das Programm bei ausbleibender Pingantwort von einer nicht existierenden IP-Adresse ausgeht. Dies ist zum Beispiel der Fall, wenn das DSP-Board über eine Internetanbindung der TU-Wien verfügt, der Computer jedoch eine andere nutzt.
Um dies zu verhindern, kann die Existenzkontrolle mittels „Ping“ anhand der Checkbox im Fenster deaktiviert werden. Dadurch ergibt sich im Fehlerfall jedoch wieder ein Zeitraum von etwa zwei Minuten bis zur entsprechenden Statusmeldung.

Soll eine serielle Verbindung aufgebaut werden, muss der richtige Com-Port und die passende Baudrate mit Hilfe zweier Popups (siehe Abbildung 2.13b) angegeben werden. Als Port kann neben einer im Rechner fix eingebauten seriellen Schnittstelle auch ein USB-RS232-Adpater dienen. Die Baudrate beträgt üblicherweise 115200Bd, kann jedoch bei Bedarf niedriger gewählt werden.
Die verfügbaren Com-Ports werden bei Öffnen des Verbindungsfensters aktualisiert. Soll die Liste aktualisiert werden, muss das Fenster neu geöffnet werden.

a)



b)

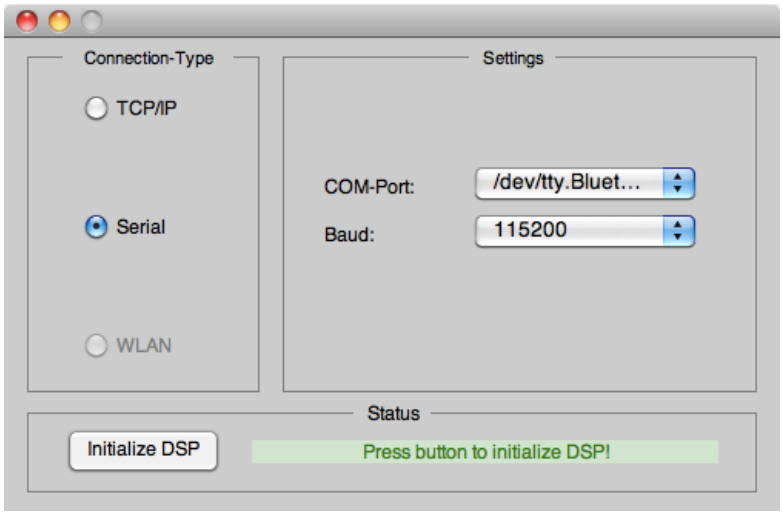


Abbildung 2.13[1]*: Fenster zur Einstellung der Verbindungsparameter*
a) TCP/IP-Verbindung
b) Serielle-Verbindung

Die Bezeichnung der verfügbaren Ports erfolgt je nach Betriebssystem unterschiedlich. Wie in Abbildung 2.13b erkennbar, werden diese in einem Unix-System direkt mit dem Namen der entsprechenden Hardware bezeichnet.
In einer Windows-Umgebung listet MATLAB die Schnittstellen im Popup unter der Bezeichnung COM1, COM2, COMx,... auf.
Zum Start der Initialisierung muss ebenfalls der Button „Initialize DSP“ betätigt werden. Während dieses Vorgangs kann, wie bereits bei der Initialisierung via TCP/IP beschrieben, das Fenster nicht geschlossen werden. Wenn dieser Ablauf erfolgreich ist, schließt sich das Fenster selbständig. Andernfalls wird mittels Statusmeldung auf einen Fehler hingewiesen.
Alle eingestellten Werte, wie Port oder IP-Adresse, werden bei Beendigung des Projekts in der Projektdatei gesichert. Somit sind diese beim nächsten Projektstart wieder bei den entsprechenden Einstellungen verfügbar.

[1] Quelle: MATLAB-GUI

2.3.5 Bereiche und Funktionen des GUI

Die Oberfläche des MATLAB-GUI ist in acht Bereiche gegliedert. Diese stellen die unterschiedlichen Funktionen des GUI dar. Die Elemente in den Bereichen 2, 3 und 4 bilden, waagrecht betrachtet, je eine zusammengehörige „Variablenzeile“.
Die Aufteilung in Funktionsgebiete und Variablenzeilen wird in Abbildung 2.14 veranschaulicht.
Das GUI ist in der Lage, bis zu 24 Variablenzeilen gleichzeitig aufzulisten. Die Begründung für maximal 24 Variablenzeilen hat zwei Ursachen: Zum einen ist eine größer Anzahl an Zeilen im GUI nicht mehr übersichtlich darstellbar. Zum anderen können, wie bereits unter Abschnitt 2.2.7.7 erläutert, bis zu acht Variablen gleichzeitig aus dem DSP angefordert werden. Ein Vielfaches von acht nützt somit die Möglichkeiten des Kommunikationsprotokolls effizient aus.
In einer Zeile werden der Name sowie alle Parameter und Werte einer ausgewählten Variable dargestellt.
Weiters entspricht jede Variablenzeile, beginnend mit der Ersten, einem fix zugewiesenem Index zwischen eins und 24. Dieser Index wird zur Index-Adressen-Zuweisung herangezogen, die in Abschnitt 2.2.5 erläutert wird. Der Index der Zeile repräsentiert somit die Variablenadresse der ihr zugewiesenen Variable.

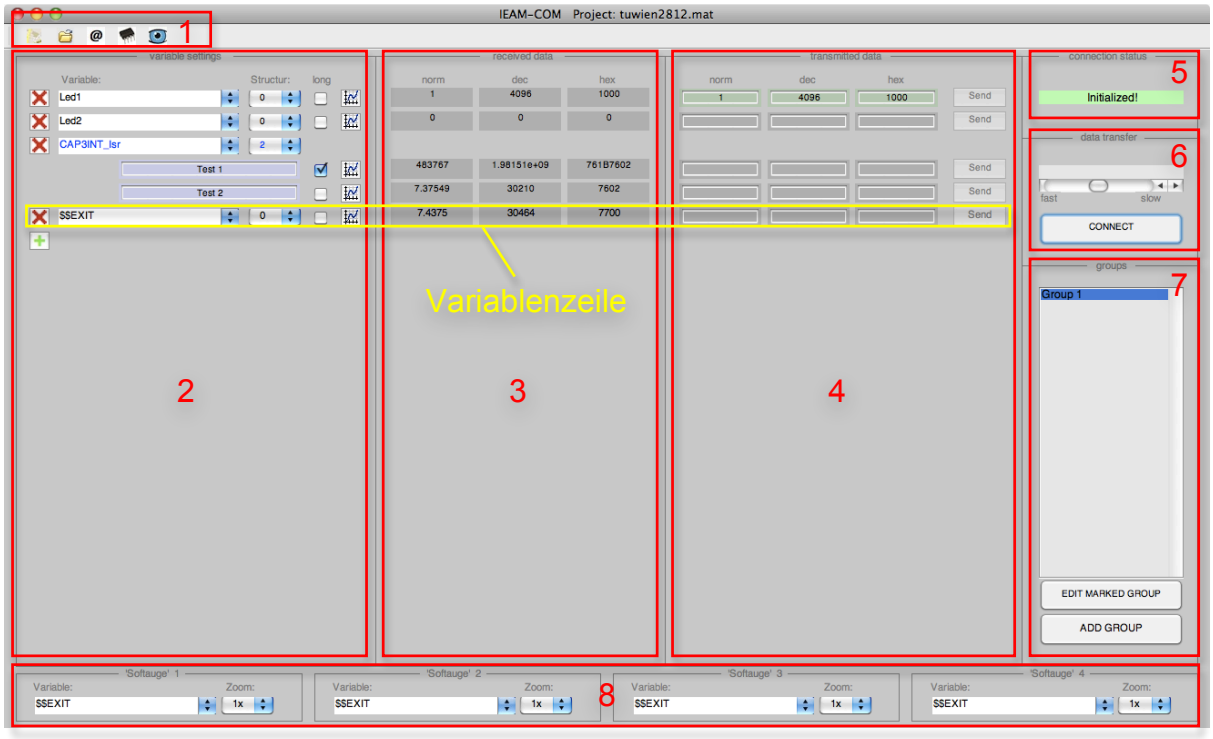


Abbildung 2.14[1]: Aufteilung des GUI in Bereiche und Variablenzeilen
Der gelb dargestellte Rahmen repräsentiert die zusammengehörigen Elemente für eine Variable ⇒ eine Variablenzeile.
Die rot umrandeten Flächen stellen die acht Bereiche des GUI dar.

[1] Quelle: MATLAB-GUI

2.3.5.1 Bereich 1: Menüleiste

Bereich 1 umfasst die Menüleiste mit ihren fünf Buttons (siehe Abbildung 2.15). Diese Knöpfe dienen dazu, die Benutzeroberfläche zu konfigurieren bzw. weitere Funktionen zu aktivieren.

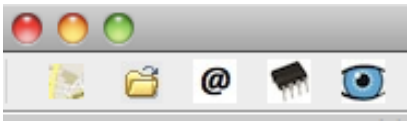


Abbildung 2.15[1]*: Bereich 1 des MATLAB-GUI*

Der Button in Form einer Landkarte dient zur Auswahl eines neuen Map-Files bzw. eines neuen Projekts, unabhängig vom Projektstart. Dieser Vorgang wird in Abschnitt 2.3.2 näher erläutert.
Der Knopf mit dem Ordner-Symbol dient zum Öffnen des Verwaltungsfensters der GUI-Variablen. Details zu diesem Fenster finden sich unter Punkt 2.3.3
Wie bereits in Teil 2.3.4 erörtert, kann man bei Betätigung des @-Symbols die Verbindungseinstellungen vornehmen und überprüfen bzw. den DSP initialisieren.
Der Button mit der Abbildung eines elektronischen Bauteils dient zum Auslesen des DSP-RAM. Details zu dieser Funktion werden in Abschnitt 2.3.6 erläutert.
Das Auge in der Menüleiste symbolisiert die Funktion „The Eye“. Bei Betätigung des Buttons wird das Fenster zur grafischen Darstellung der empfangenen Daten sichtbar. Bei erneuter Betätigung des Auges wird das Fenster wieder unsichtbar. Die vorgenommenen Einstellungen bleiben jedoch erhalten. Ebenso werden laufende Aufnahmen von Daten durch die erneute Betätigung nicht unterbrochen. Der genaue Funktionsumfang von „The Eye“ wird unter Punkt 2.3.7 dargestellt.

2.3.5.2 Bereich 2: Variablen-Einstellungen

Bereich 2 des GUI dient der Auflistung von bis zu 24 Variablen und deren spezifische Einstellungen. Abbildung 2.16 zeigt, wie dieser Bereich prinzipiell aussehen kann.
Durch Betätigen des roten „X“ kann die Variable der betreffenden Zeile gelöscht werden. Sind im GUI noch keine 24 Zeilen aufgelistet, kann durch anklicken des grünen „+“-Symbols eine Weitere hinzugefügt werden.
Die zur Kommunikation mit dem DSP nötigen Adressen der GUI-Variabeln, dargestellt durch den Variablennamen, und ihre zugewiesenen Indizes werden im Hintergrund vom MATLAB-GUI selbständig verwaltet und nicht explizit angezeigt.
Die Namen der gewählten GUI-Variaben werden in einem Popup dargestellt. Dieses Popup enthält eine Liste aller Map-File-Variablen und kann durch Betätigen der blauen Pfeiltasten geöffnet werden. In der Liste des Popups ist stets die gewählte Variable markiert. Will man eine andere Variable im GUI darstellen, so muss man diese nur aus der Liste des entsprechenden Popups auswählen. Mit der Escape-Taste kann das Popup, ohne Änderung der Variablenliste, wieder geschlossen werden.

[1] Quelle: Ausschnitt aus dem MATLAB-GUI

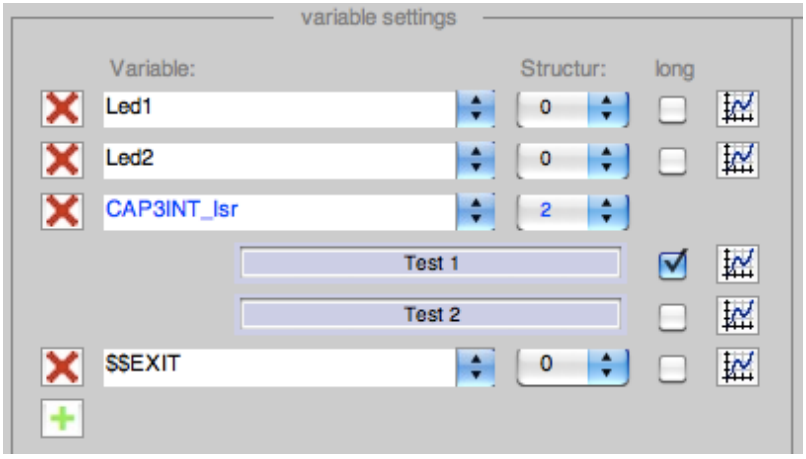


Abbildung 2.16[1]*: Prinzipielle Darstellung von Bereich 2 des MATLAB-GUI*

Um die Suche nach einer Variable zu vereinfachen, kann auch hier ein Suchstring vorgegeben werden. Die eingegebene Zeichenfolge wird in diesem Fall oberhalb des Popups dargestellt und die Liste der Map-File-Variablen wird auf jene Variablen eingeschränkt, deren Name mit dem eingegebenen String beginnen.
Wird eine neue Variable mit Hilfe des Popups ausgewählt, so wird automatisch im Anschluss ein Fenster zur Abfrage des Datentyps der neuen Variable geöffnet. Dieses Fenster ist in Abbildung 2.17 dargestellt.

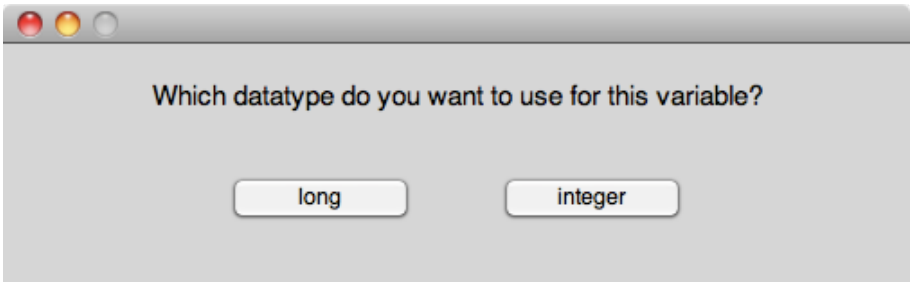


Abbildung 2.17[2]*: Fenster zur Abfrage des Datentyps*

Durch Anklicken des entsprechenden Datentyps wird dieser übernommen und das Fenster geschlossen. Falls man dieses Fenster mit der Eingabe von „Enter“ bestätigt, wird der in den Software-Parametern vorgegebene Standardwert (siehe Abschnitt 2.3.8) übernommen.
Den ausgewählten Datentyp repräsentiert eine Checkbox in der jeweiligen Variablenzeile. Wird diese markiert, stellt sie eine Variable vom Typ Long dar. Ohne Markierung werden die Variablenwerte als Integerdaten behandelt. Durch Änderung des Checkboxstatus kann der Datentyp jederzeit angepasst werden.
Weshalb eine diesbezügliche Unterscheidung notwendig ist, soll die folgende Erläuterung zeigen.

---

[1] Quelle: Ausschnitt aus dem MATLAB-GUI
[2] Quelle: MATLAB-GUI

Der DSP ist in der Lage, Variablen als Long oder Integer zu verwalten. Je nach Datentyp sind somit, wie bereits unter Abschnitt 2.2.3 dargestellt, für die Darstellung des gesamten Wertebereichs zwei bis vier Bytes pro Variable notwendig.
Die Typendeklaration der Variablen erfolgt in der Firmware des Prozessors. Diese Deklaration sorgt für eine Reservierung der benötigten Anzahl an Bytes im Speicher des DSP.
Jeder Speicherplatz im DSP ist über eine fixe Adresse erreichbar. Dabei werden jeweils zwei Byte, ein Low- und ein High-Byte, zu einer Adresse zusammengefasst. Eine als Integer deklarierte Variable kann somit direkt einem Speicherplatz zugewiesen werden und ist dadurch über eine Adresse erreichbar.
Soll eine als Long deklarierte Variable im Speicher des DSP abgelegt werden, sind vier Bytes notwendig. Diese werden durch zwei Adressen beschrieben, die direkt nebeneinander liegen. Die Daten der Longvariable werden derart abgelegt, dass der niedrigere Adresswert die in der Wertigkeit niedrigeren Bytes beschreibt (Variante „Little Endian“). Durch eine Inkrementierung dieser Adresse kann auf die beiden höherwertigen Bytes zugegriffen werden. Somit ist es möglich, eine im Speicher abgelegte Longvariable über die niedrigere der beiden notwendigen Adressen zu erreichen.
Eine schematische Darstellung der Adressierung und der Aufteilung der einzelnen Bytes zeigt Abbildung 2.18.

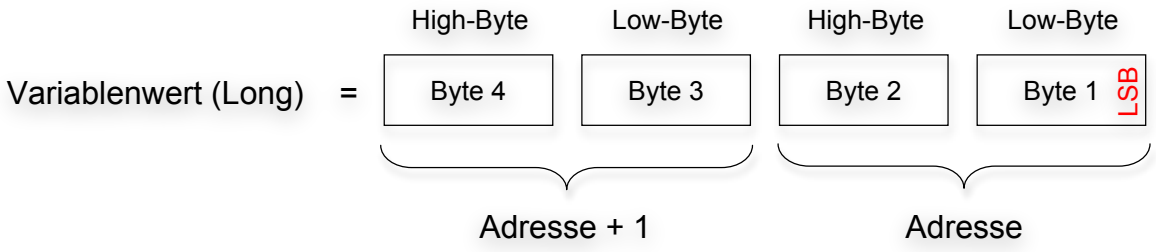


Abbildung 2.18: *Adressierungsschema der Variablenbytes*
Der gesamte Variablenwert ergibt sich durch einfaches aneinanderreihen der einzelnen Bytes.
LSB symbolisiert das Bit mit dem niedrigsten Wert in der dargestellten Variable.

Beim Kompilieren der DSP-Firmware wird automatisch ein Map-File erzeugt. Dieses Map-File enthält, wie bereits unter Abschnitt 2.3.1 erläutert, eine Auflistung aller Variablen und deren zugehörige Adressen. Diese Adressen weisen, wie oben erläutert, den Weg zum Speicherplatz der Variablen im DSP. Eine Information über den Datentyp ist in dieser Auflistung jedoch nicht vorhanden.
Soll nun durch das GUI eine Variable ausgelesen oder geschrieben werden, so wird diese indirekt (siehe Abschnitt 2.2.5) mit Hilfe der entsprechenden Adresse im DSP-Speicher identifiziert. Zusätzlich wird, neben der Lese- bzw. Schreibanforderung, auch der in das GUI eingetragene Datentyp der Variable an den DSP übermittelt.

Empfängt der DSP eine Anforderung für eine Integervariable, so werden die beiden der Adresse entsprechenden Bytes direkt beschrieben oder in das GUI übertragen. Bei einer Anforderung für eine Longvariable wird vom DSP die Adresse zusätzlich inkrementiert, um auf einen Bereich von vier Bytes zugreifen zu können. Diese vier Bytes werden in weiterer Folge, je nach Bedarf, an das GUI gesendet oder verändert.
Durch diese Darstellung kann man erkennen, dass eine falsche Angabe des Datentyps zu einem falschen Variablenwert führen kann. Soll beispielsweise eine Integer-Variable überschrieben werden, die im GUI fälschlicherweise als Long-Datentyp definiert wurde, so werden vom DSP durch die Inkrementierung der Adresse vier Bytes im DSP-Speicher verändert. Von diesen vier Bytes sind jedoch die beiden Letzten bereits Daten einer anderen Variable. Umgekehrt, wird eine Long-Variable im GUI als Integer markiert und ausgelesen, werden nur zwei der benötigten vier Bytes übermittelt. Beide Fälle müssen vermieden werden, um die Datenintegrität zu wahren.
Eine Besonderheit dieser Kommunikationssoftware stellt die Möglichkeit dar, Strukturen zu verwalten. Als Strukturen werden Variablen bezeichnet, die aufgrund einer Gemeinsamkeit zusammengefasst werden. Beispielsweise können die beiden Variablen „Name“ und „Geburtstag“ dem Überbegriff „Person“ zugeordnet werden. Dieser Überbegriff an sich stellt wieder eine Variable dar, in der die beiden anderen Variablen enthalten sind. Wie viele Strukturlemente zu einer übergeordneten Variable, im Zuge dieser Arbeit als „Structurparent“ bezeichnet, zusammengefasst werden, bestimmt man je nach Bedarf.
Theoretisch kann jedes Strukturelement wieder verzweigt werden. Dies wäre jedoch im MATLAB-GUI nicht übersichtlich darstellbar und ist somit nicht vorgesehen.
Strukturen werden, so wie alle Variablen des MATLAB-GUI, in der Firmware des DSP deklariert. Im Zuge des Flashvorgangs werden im DSP, entsprechend der Anzahl an Strukturelementen und der jeweiligen Datentypen, die Menge an benötigten Bytes bzw. Adressen reserviert. Die reservierten Adressen für eine Struktur kommen „nebeneinander“ zu liegen. Dabei entspricht die Adresse mit dem kleinsten Wert dem Speicherort der ersten Strukturvariable. Jede weitere Strukturvariable kann durch diese Adresse, unter Beachtung der einzelnen Datentypen, beschrieben werden. Dabei gilt, dass die Anschrift eines Elements vom Typ Long zweimal inkrementiert werden muss, um die Adresse der nächsten Strukturvariable darzustellen. Im Fall einer Integervariable reicht eine einfache Inkrementierung der Adresse aus, um das folgende Element zu beschreiben.
Zur Veranschaulichung dieses Adressierungsvorgangs sei auf das Beispiel in Abbildung 2.19 verwiesen.
Bei der automatischen Erstellung des Map-Files werden Strukturen jedoch nicht vollständig erfasst. Es werden in dieser Datei Strukturen nur jeweils mit dem Structurparent und der Adresse des ersten Strukturelements aufgelistet. Durch das Wissen, wie die einzelnen Strukturvariablen adressiert sind, kann aber trotzdem mit dem GUI auf die einzelnen Elemente zugegriffen werden.
Ist einem bewusst, welche Variable im Map-File eine Struktur darstellt, kann man diese als normale Variable auswählen. In weiterer Folge muss man mit Hilfe des Popups in der Rubrik „Structur“ die Anzahl der definierten Strukturvariablen auswählen. Dadurch ändert sich, wie in Abbildung 2.16 dargestellt, das Erscheinungsbild des MATLAB-GUI.

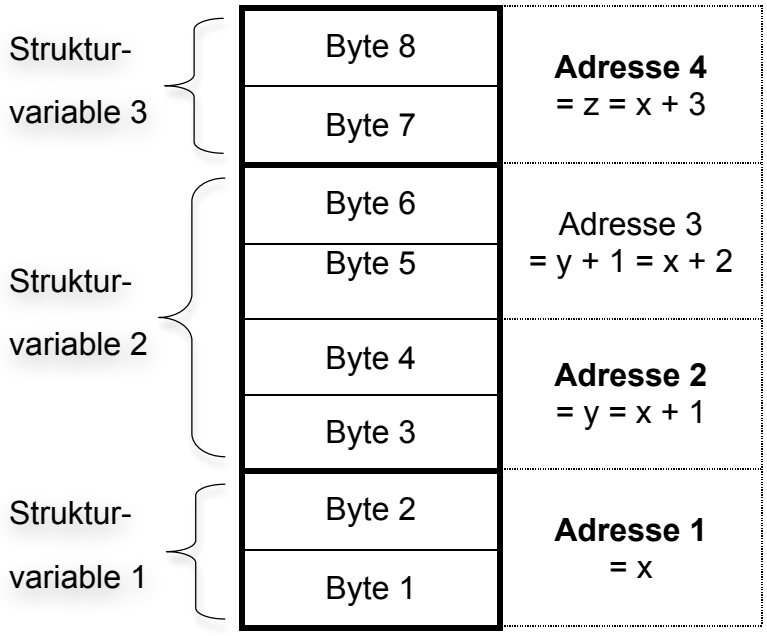


Abbildung 2.19: Beispiel zur Adressierung von Strukturvariablen
Variable 1 und 3 sind vom Typ Integer, Variable 2 ist vom Typ Long.
Die in der Abbildung fett markierten Einträge entsprechen der Adresse der jeweiligen Strukturvariable. Die Einträge „x“, „y“ und „z“ repräsentieren dabei den Wert der jeweiligen Adresse.

Das im Map-File aufgelistete Structurparent wird im GUI blau markiert und alle Schreib- bzw. Lesefelder werden in dieser Variablenzeile entfernt. Unter dieser Variablenzeile werden Zeilen für die Strukturvariablen eingefügt, deren Adressen vom GUI automatisch im Hintergrund erstellt werden. Durch die Änderung des Datentyps einer Variable werden automatisch alle betreffenden Adressen der Strukturelemente angepasst. Dabei wird nach dem oben erläutertem und in Abbildung 2.19 dargestellten Prinzip vorgegangen. Der Name jedes Elements kann jederzeit im blau markierten Feld angepasst werden.
Da die Adressen der Strukturvariablen nicht im Map-File aufgelistet sind, können diese vom GUI nicht kontrolliert werden. Um trotzdem die Datenintegrität zu wahren, muss man bei der Auswahl der Anzahl der Strukturelemente und des Datentyps sehr gewissenhaft vorgehen!
In einer Variablenzeile befindet sich neben der Datentyp-Checkbox ein Button zur Auswahl eines Softaugekanals. Dieser ist erkennbar an der symbolischen Darstellung einer Funktion.
Als weiteres Element in Bereich 2 befindet sich in jeder Zeile einer aufgelisteten Variable ein Button zur Zuweisung an einen Softauge-Kanal. Dieser trägt symbolisch eine grafische Darstellung einer Funktion und befindet sich neben der Checkbox zur Auswahl des Datentyps.

Wird dieser Button betätigt, öffnet sich in unmittelbarer Umgebung ein Fenster. In diesem Fenster befindet sich eine Auflistung von vier Softauge-Kanälen. Wird einer dieser Kanäle ausgewählt, erfolgt eine analoge Ausgabe der betreffenden Variable auf dem entsprechenden Kanal. Erfolgt keine Auswahl, schließt sich das Fenster selbständig nach ein paar Sekunden. Details über die Softauge-Funktion finden sich in der Beschreibung von Bereich 8.

2.3.5.3 Bereich 3: Empfangene Daten

Bereich 3 des MATLAB-GUI umfasst die empfangenen Werte der aufgelisteten Variablen. Besteht eine Verbindung zum Datentransfer zwischen DSP und GUI, werden die Werte der GUI-Variablen in einem vorgegebenen Intervall aktualisiert.
Die empfangenen Variablenwerte werden für jede Variable normiert, dezimal und hexadezimal dargestellt (siehe Abbildung 2.20).
Der Normierungswert ist auf dezimal 2^{12} = 4096 bzw. hexadezimal 1000 festgelegt. Dieser Wert ist jedoch in den Software-Parametern (siehe Abschnitt 2.3.8) veränderbar.
Der Wert in der Rubrik „hex“ entspricht dabei der hexadezimalen Darstellung des binären Werts im DSP-Speicher. Der binäre Wert wird im DSP in Zweierkomplementdarstellung abgelegt um auch negative Daten verarbeiten zu können. Näheres zur Zweierkomplementdarstellung wird in den Erläuterungen zu Bereich 4 (Gesendete Daten) dargestellt.
Die Verbindung zum Datentransfer zwischen DSP und GUI sowie die Aktualisierungsrate werden im Bereich 6 des GUI gestartet bzw. eingestellt.

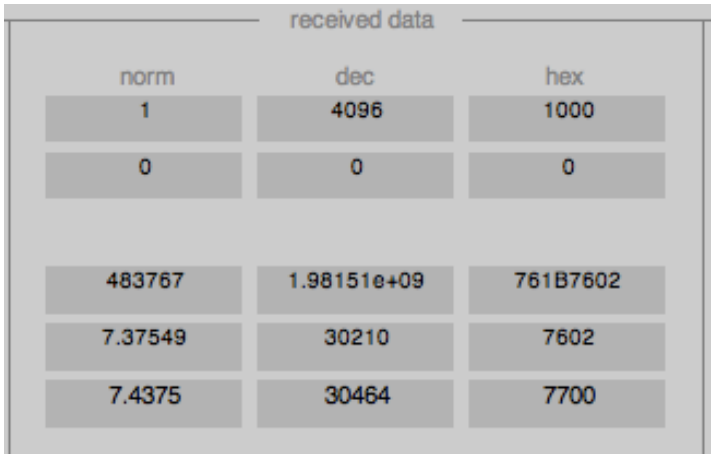


Abbildung 2.20[1]: Prinzipielle Darstellung von Bereich 3 des MATLAB-GUI

[1] Quelle: Ausschnitt aus dem MATLAB-GUI

2.3.5.4 Bereich 4: Gesendete Daten

Bereich 4 umfasst die zu sendenden bzw. gesendeten Variablenwerte. Wie in Abbildung 2.21 erkennbar, werden auch diese Werte normiert, dezimal und hexadezimal dargestellt.

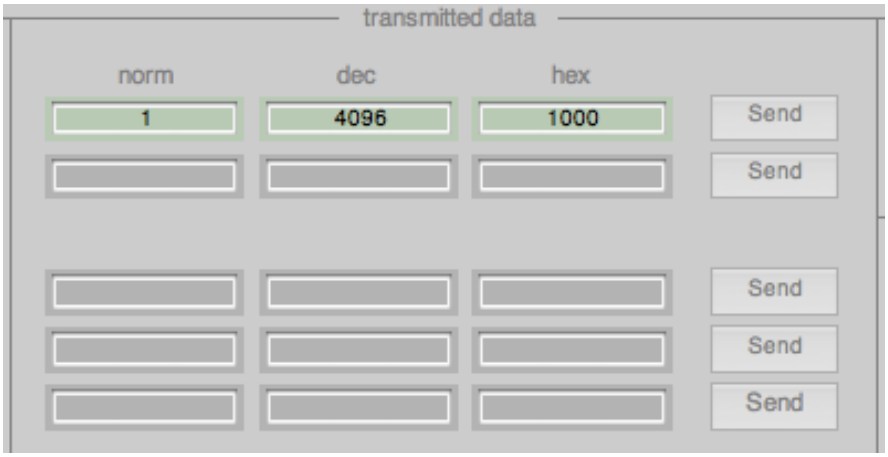


Abbildung 2.21[1]: Prinzipielle Darstellung von Bereich 4 des MATLAB-GUI

Gibt man in einem beliebigen Feld einen korrekten Wert ein, so wird dieser automatisch für die beiden anderen Felder berechnet und entsprechend eingefügt. Dabei kommt es gegebenenfalls zu einer Rundung, da ausschließlich ganze Zahlen an den DSP übermittelt werden können.
Einzig der Normwert kann mit Nachkommastellen versehen werden. Andernfalls wird das Produkt aus eingetragenem Normwert & Normierungswert vom GUI auf eine ganze Zahl gerundet und daraus automatisch der passende Normwert berechnet.
Wird ein ungültiger Wert eingegeben, beispielsweise ein Buchstabe im Dezimalbereich, akzeptiert das GUI die Eingabe nicht und stellt den zuletzt gültigen Zahlenwert wieder dar.
Handelt es sich bei dem eingegebenen Wert um eine Zahl, die außerhalb des darstellbaren Datenbereichs liegt, wird man mit einer Meldung darauf hingewiesen (siehe Abbildung 2.22). Zugleich ersetzt das GUI ebenfalls die Eingabe durch den zuletzt gültigen Wert.

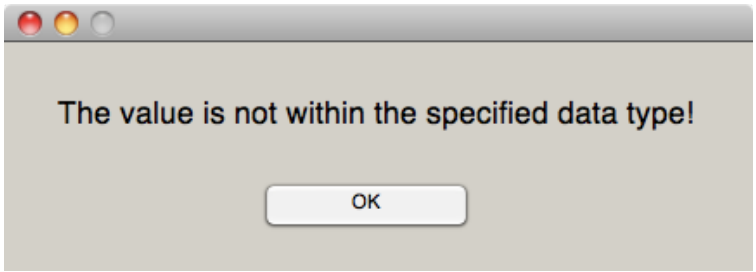


Abbildung 2.22[1]: Hinweis auf Wert außerhalb des Datenbereichs

[1] Quelle: MATLAB-GUI

Die Größe des darstellbaren Datenbereichs hängt davon ab, welcher Datentyp in der Variablenzeile anhand der Checkbox (siehe Bereich 2) ausgewählt wurde.
Tabelle 2.2 zeigt den Bereich der Zahlenwerte, die je nach Long- oder Integer-Typ in das entsprechende Feld eingetragen werden können.

	Integer			Long		
	negativ		positiv	negativ		positiv
Normierter Wert	-8 bis -0,00024414	0	0,00024414 bis 7,9998	-524.288 bis -0,00024414	0	0,00024414 bis 524.287,9998
Dezimalwert	-32.768 bis -1	0	1 bis 32.767	-2.147.483.648 bis -1	0	1 bis 2.147.483.647
Hexadezimalwert	8000 bis FFFF	0	1 bis 7FFF	80000000 bis FFFFFFFF	0	1 bis 7FFFFFFF

Tabelle 2.2: Übersicht über die darstellbaren Wertebereiche

Wie in Tabelle 2.2 zu erkennen, können sowohl positive als auch negative Werte verarbeitet werden. Um negative Zahlen im Binärsystem darstellen zu können, sind die Variablen im DSP in Zweierkomplementdarstellung abgelegt.
Der Hexadezimalwert im GUI entspricht dabei genau diesem binären Zweierkomplement im DSP-Speicher. Aus diesem Grund beinhalten in Tabelle 2.2 die negativen Hex-Werte keine Vorzeichen.
Eine positive Zahl wird in der Zweierkomplementdarstellung durch ein „Most Significant Bit“ (MSB) mit dem Wert 0 symbolisiert. Im Gegensatz dazu signalisiert eine 1 als MSB eine negative Zahl. Dies führt, im Vergleich zur normalen binären Darstellung, zu einer Verschiebung des darstellbaren Wertebereichs in den negativen Zahlenbereich. Es kann dadurch auf eine zusätzliches Bit zur Darstellung des Vorzeichens verzichtet werden.
Tabelle 2.3 zeigt eine Gegenüberstellung der darstellbaren Werte einer vier Bit großen Zahl in Zweierkomplement- und in binärer Darstellung.

Binäre Darstellung			Zweierkomplementdarstellung		
Dezimalwert	Binärwert	Hexadezimalwert	Dezimalwert	Binärwert	Hexadezimalwert
15	1111	F			
14	1110	E			
13	1101	D			
12	1100	C			
11	1011	B			
10	1010	A			
9	1001	9			
8	1000	8			
7	0111	7	+7	0111	7
6	0110	6	+6	0110	6
5	0101	5	+5	0101	5
4	0100	4	+4	0100	4
3	0011	3	+3	0011	3
2	0010	2	+2	0010	2
1	0001	1	+1	0001	1
0	0000	0	0	0000	0
			-1	1111	F
			-2	1110	E
			-3	1101	D
			-4	1100	C
			-5	1011	B
			-6	1010	A
			-7	1001	9
			-8	1000	8

Tabelle 2.3: Vergleich der binären Darstellung und des Zweierkomplements

Insgesamt können mit einer vier Bit großen Zahl 16 unterschiedliche Werte repräsentiert werden.
Der in der Tabelle aufgeführte Hexadezimalwert entspricht der direkten Umrechnung des aufgelisteten Binärwerts. In dieser hexadezimalen Form wird der Variablenwert u.a. auch im GUI dargestellt.
Deutlich zu erkennen ist, dass im Zweierkomplement der Datenbereich in den negativen Bereich verschoben wird (rot markiert). Die Null wird der positiven Hälfte (blau markiert) zugerechnet.

Das Zweierkomplement kann nach dem folgenden Schema berechnet werden:

- Handelt es sich um eine **positive Zahl**, kann diese einfach in das Binärsystem umgewandelt werden und bedarf keiner weiteren Behandlung.

- Soll eine **negative Zahl** in die Zweierkomplementdarstellung umgerechnet werden, muss deren Vorzeichen ignoriert und der Betrag in das Binärsystem umgeformt werden.

- In weitere Folge wird diese binäre Zahl invertiert.

- Zum Abschluss muss zur invertierten Zahl 1 addiert werden. Ein gegebenenfalls auftretender Überlauf (die n+1. Stelle einer n-Bit breiten Zahl) ist zu vernachlässigen!

- Wesentlich bei der Umrechnung in das Zweierkomplement ist, dass sich die Zahl im zulässigen Wertebereich des jeweiligen Datentyps befindet (siehe Tabelle 2.2)!

- Z.B.: Umrechnung in eine 8-Bit breite Zweierkomplementdarstellung

  Positive Zahl: $+48_{dezimal} = 00110000_{zweierkomplement = binär}$

  Negative Zahl: $-80_{dezimal}$

  1. Betrag Umwandeln $\Rightarrow +80_{dezimal} = 01010000_{binär}$

  2. Invertieren von 01010000 $\Rightarrow$ 10101111

  3. Addieren von 1 $\Rightarrow$ 10101111 + 1 = 10110000

  $\Rightarrow -80_{dezimal} = 10110000_{zweierkomplement}$

Eine Umrechnung einer Zweierkomplementdarstellung in eine Dezimalform folgt der umgekehrten Vorgehensweise:

- Ist die **erste Stelle 0**, handelt es sich um eine **positive Zahl**. Diese kann direkt aus dem binären System in eine Dezimaldarstellung umgeformt werden.

- Eine **1 an der ersten Stelle** symbolisiert eine **negative Zahl**. Dies bedeutet, dass der dargestellten Zahl 1 subtrahiert werden muss und im Anschluss die einzelnen Ziffern zu negieren sind.

- In weitere Folge wird diese binäre Zahl in das Dezimalsystem umgerechnet und mit einem negativen Vorzeichen versehen.

Die Übertragung des eingetragenen Werts kann schließlich auf zwei Arten erfolgen. Entweder betätigt man den „Send-Button“ der jeweiligen Variablenzeile oder die Eingabe eines Werts wird mit der Eingabe von „Enter“ abgeschlossen.
Für beide Fälle wird vorausgesetzt, dass zuvor die Verbindung zum Datentransfer zwischen DSP und GUI hergestellt worden ist (siehe Bereich 6 des GUI). Andernfalls sind die „Send-Buttons“ deaktiviert bzw. wird eine Eingabe von „Enter“ vom GUI ignoriert.
Im Zuge eines Projektes kann es sinnvoll sein, die selbe Variable mehrmals im GUI aufzulisten. Beispielsweise kann ein Freigabeparameter zweimal aufgelistet und einmal mit dem „Start-“ und einmal mit dem „Stopwert“ versehen werden. In weiterer Folge muss nur noch der entsprechende Sende-Button betätigt werden, ohne den Wert ändern zu müssen.
Um jedoch erkennen zu können, welcher dieser Werte zuletzt an den DSP übermittelt wurde, wird dieser nach erfolgreicher Versendung grün hinterlegt. Alle anderen „Transmitted“-Werte der gleichen Variable werden in einem grauen Hintergrund dargestellt.
Diese Einfärbung gilt auch für einmalig aufgelistete Variablen. Wird der zu sendende Wert in das GUI eingetragen, jedoch noch nicht übermittelt, ist der Hintergrund grau gefärbt. Erst nach erfolgreicher Übertragung färbt sich dieser grün (siehe Abbildung 2.21).

2.3.5.5 Bereich 5: Verbindungsstatus

In Bereich 5 wird signalisiert, ob die Verbindung zum DSP in Ordnung ist und dieser bereits initialisiert wurde.

Abbildung 2.23 zeigt die unterschiedlichen Statusmeldungen. Wird der Status „Not initialized!“ dargestellt, erfolgte entweder noch keine Initialisierung des DSP oder es ist ein Fehler beim Datentransfer aufgetreten. In beiden Fällen muss eine erneute Initialisierung (siehe Abschnitt 2.3.4) erfolgen, um eine Verbindung zum Datentransfer (siehe Bereich 6 des GUI) zu ermöglichen.

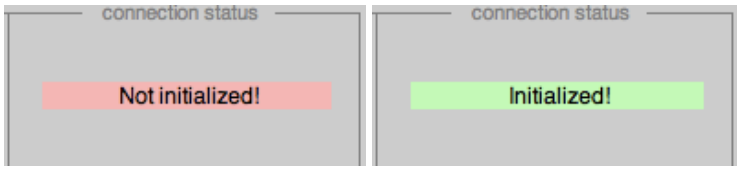


Abbildung 2.23[1]*: Statusmeldungen für Verbindung*

[1] Quelle: Ausschnitt aus dem MATLAB-GUI

2.3.5.6 Bereich 6: Verbindung für Datentransfer

Wie bereits in der Einleitung erläutert, dient das MATLAB-GUI der Parametrierung eines Umrichters. Im Zuge der Parametrierung muss man einen Überblick darüber erhalten, wie sich einzelne Variablen im Betrieb verhalten. Außerdem muss man erkennen können, wie sich die Änderung einer Variable auf die restlichen Parameter auswirkt.

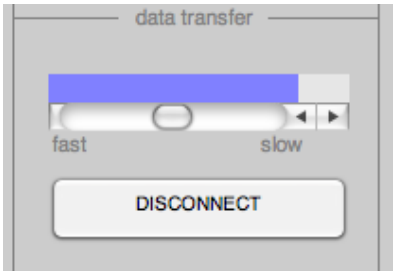


Abbildung 2.24[1]: Darstellung der Aktualisierungsrate

Um dies zu ermöglichen, kann der Benutzer in Bereich 6 des GUI eine Verbindung für den Datentransfer zwischen DSP und GUI aktivieren.
Wie bereits unter Abschnitt 2.2 erläutert, bildet der Computer den Master und der DSP den Slave. Sollen Variablenwerte aus dem DSP ausgelesen werden, müssen diese vom Master explizit angefordert werden. Diese Anforderung an den DSP wird vom GUI bei aktivierter Datentransfer-Verbindung für alle aufgelisteten Variablen laufend durchgeführt.
Die Aktualisierungsrate, in der die jeweils aktuellen Variablenwerte vom DSP angefordert werden, kann anhand des Schiebereglers oberhalb des „Connect-Buttons“ eingestellt werden. Diese Rate wird jedoch auch von der Verbindungs-qualität zwischen DSP und Computer beeinflusst.
Abbildung 2.24 zeigt diesen Bereich des GUI. Der im Bild dargestellte Button „Disconnect“ ändert seine Aufschrift abhängig vom Status und wird in weiterer Folge als „Connect/Disconnect-Button“ bezeichnet.
Als Voraussetzung für die Datentransfer-Verbindung gilt eine korrekte Initialisierung des DSP. Dies wird anhand des Status mit der Meldung „Initialized!“ im Bereich 5 des GUI signalisiert. Durch das Betätigen des „Connect/Disconnect-Buttons“ wird dem DSP die aktuelle Index-Adressen-Zuweisung (Erläuterung siehe Abschnitt 2.2.5) sowie die entsprechenden Adressen und Skalierungsfaktoren der Softauge-Kanäle übermittelt. Im Anschluss wird der laufende Datentransfer gestartet.
Direkt über dem Schieberegler befindet sich ein blauer Statusbalken, der die eingestellte Aktualisierungsrate wiedergibt. Jeder Erweiterungsschritt des Balken symbolisiert einen vollständigen Aktualisierungsvorgang aller aufgelisteten Variablen. Kommt es dabei zu unterschiedlich schnellen Schritten, deutet das auf eine schlechte Verbindungsqualität hin.
Soll der laufende Datentransfer unterbrochen werden, muss abermals der Button „Connect/Disconnect“ betätigt werden.

[1] Quelle: Ausschnitt aus dem MATLAB-GUI

Kommt es während dem Datentransfer zu einem Verbindungsfehler, wird die Verbindung getrennt, der Status in Bereich 5 des GUI auf „Not initialized!“ geändert und eine Meldung ausgegeben (siehe Abbildung 2.25).

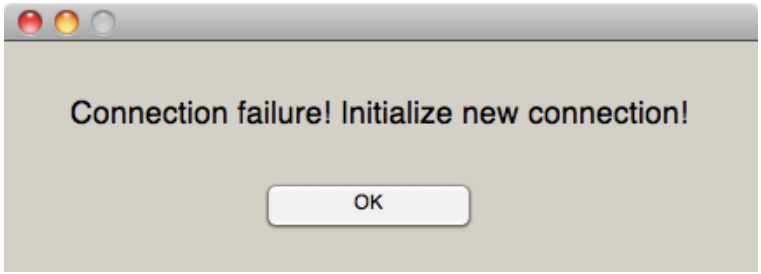


Abbildung 2.25[1]*: Hinweis auf Verbindungsfehler*

Wie bereits unter Punkt 2.3.2 ausführlich dargestellt, ist beim Start eines Projekts ein Map-File auszuwählen, dessen Pfad im GUI hinterlegt ist.
Wird die Firmware des DSP verändert, muss diese im Prozessor neu gespeichert (geflasht) werden. Dabei wird auch ein neues Map-File generiert, da sich die Adressen und die Anzahl der Variablen mit der neuen Firmware ändern können.
Um sicherzustellen, dass dem GUI stets eine aktuelle Variablen- und Adressauflistung vorliegt, wird bei jedem Start der Datentransfer-Verbindung das Map-File anhand des hinterlegten Pfads erneut geladen. Im Zuge dessen werden alle im GUI aufgelisteten Variablen, ausgenommen die Strukturvariablen, kontrolliert, ob diese im neu geladenen Map-File noch vorhanden sind. Sind die aufgelisteten Variablen in Ordnung, erfolgt eine erneute Zuweisung der entsprechenden Adressen. Die Adressen der Strukturvariablen werden, basierend auf der neu zugewiesenen Adresse der Structurparent-Variable, durch Berechnung erneuert.
Stellt sich bei der Kontrolle heraus, dass eine im GUI aufgelistete Variable im neu geladenen Map-File nicht mehr vorhanden ist, so wird diese rot markiert. Außerdem erscheint ein entsprechendes Hinweisfenster (siehe Abbildung 2.26). Ein Datentransfer kann erst nach Beseitigung der ungültigen Variablen erfolgen.
Aus diesem Grund ist es sinnvoll, das Map-File an jenem Speicherort zu belassen, an dem beim Flashvorgang stets automatisch die aktuellste Datei abgelegt wird.

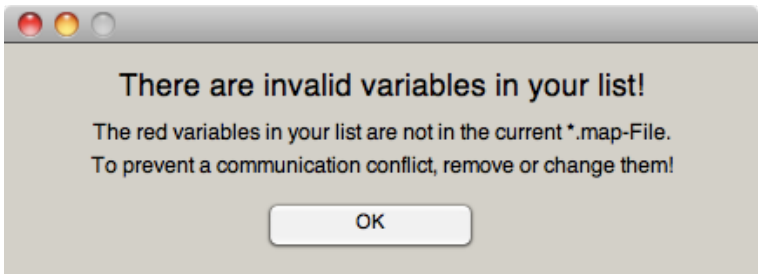


Abbildung 2.26[1]*: Hinweis auf ungültige Variablen im GUI*

[1] Quelle: MATLAB-GUI

Sollte das bereits vom GUI eingelesene Map-File gelöscht oder verschoben worden sein, so kann dieses beim Start der Datentransfer-Verbindung nicht mehr geladen werden. Dies wird einem durch eine Meldung, dargestellt in Abbildung 2.27, mitgeteilt. Abhilfe schafft nur, das Map-File wieder an die ursprüngliche Position zu verschieben oder ein neues Projekt anzulegen.

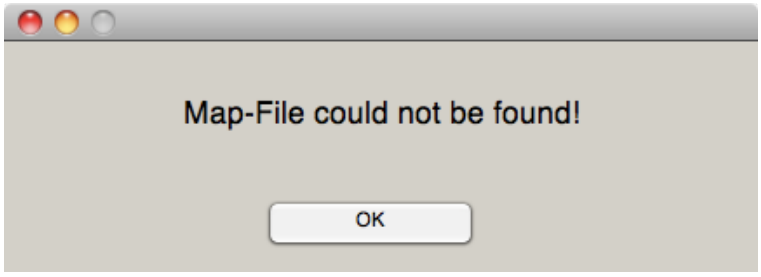


Abbildung 2.27[1]: Hinweis auf fehlendes Map-File

Ist die Datentransfer-Verbindung aktiv und ändert man die Variablenauflistung (z.B. ändern oder löschen einer Variable), wird vom GUI eine aktualisierte Index-Adressen-Zuweisung (siehe Abschnitt 2.2.5) an den DSP gesendet!
Auch die Änderung eines Datentyps einer Strukturvariable kann Einfluss auf die Adressen der GUI-Variablen haben, wodurch bei Bedarf ebenfalls eine neue Index-Adressen-Zuweisung an den DSP übermittelt wird.

2.3.5.7 Bereich 7: Gruppen

Bereich 7 des MATLAB-GUI umfasst die Gruppenverwaltung eines Projekts. Durch Gruppen erhält man die Möglichkeit, Variablen vom gleichen Typ (z.B. Ströme oder Spannungen) zusammenzufassen. Weiters ist man in der Lage, in jeder Gruppe bis zu 24 Variablen aufzulisten, wodurch ein Projekt beliebig viele Variablen umfassen kann.
Wird ein neues Projekt gestartet, generiert das GUI automatisch die Gruppe „Group 1“, in der alle zu Beginn ausgewählten Variablen (siehe Abschnitt 2.3.3) abgelegt werden.
Soll im Laufe des Projekts eine neue Gruppe erstellt werden, muss der Button „ADD GROUP“ betätigt werden (siehe Abbildung 2.28). Dadurch sichert das GUI alle bis zu diesem Zeitpunkt aufgelisteten Variablen inklusive aller Daten in der bisherigen Gruppe und stellt im Anschluss eine leere GUI-Oberfläche dar.
Diese leere Oberfläche stellt die neue Gruppe dar. Anhand des Variablenfensters, welches durch den in Abbildung 2.11 gezeigten Button geöffnet wird, können nach Bedarf neue Variablen eingefügt werden.
Ein Gruppenwechsel erfolgt über Auswahl der entsprechenden Gruppe in der Gruppenliste (siehe Abbildung 2.28). Alle Variablendaten im GUI werden vor dem Laden der neu gewählten Gruppe gesichert und stehen somit bei Bedarf wieder zur Verfügung.

[1] Quelle: MATLAB-GUI

Bei Betätigung des „EDIT MARKED GROUP“ Button öffnet sich ein Fenster zum kopieren, umbenennen oder löschen der blau markierten Gruppe. Wird keine der aufgelisteten Optionen gewählt, schließt sich das Fenster nach einigen Sekunden wieder.

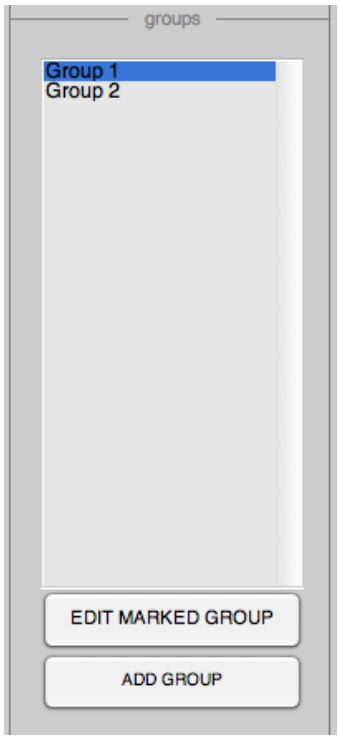


Abbildung 2.28[1]: Darstellung der Gruppenliste

Das Löschen einer Gruppe ist nur möglich, wenn mehr als eine Gruppe in der Liste aufgeführt wird.
Das Kopieren einer Gruppe kann vorteilhaft sein, wenn sich Gruppen nur in einigen Variablen unterscheiden.
Wird während einer aktivierten Datentransfer-Verbindung eine Gruppe gewechselt, aktualisiert das GUI im DSP automatisch die Index-Adressen-Zuweisung der Variablen aus der gewählten Gruppe und hält die Verbindung aktiv. Wird jedoch während der aktiven Verbindung auf eine leere Gruppe gewechselt oder eine neue Gruppe hinzugefügt, deaktiviert das GUI die Datentransfer-Verbindung, da keine zu aktualisierenden Variablen im GUI zur Verfügung stehen.

[1] Quelle: Ausschnitt aus dem MATLAB-GUI

2.3.5.8 Bereich 8: Softauge

Abbildung 2.29 zeigt den ersten von vier Kanälen des „Softauges“. Die Softauge-Funktion ermöglicht die analoge Ausgabe des normierten Wertes von vier verschiedenen Variablen auf dem DSP-Board.
Mit einem entsprechenden Adapter, der an das DSP-Board angeschlossen werden muss, ist es möglich, die vier Kanäle über je ein BNC-Kabel auf ein Oszilloskop zu übertragen.
Durch die analoge Ausgabe der Signale am DSP-Board, ist eine örtliche Gebundenheit der Softauge-Funktion an den Standort des Boards gegeben!
Anhand der vier in Bereich 8 integrierten Popups kann die gewünschte Variable für den entsprechenden Softauge-Kanal ausgewählt werden. Weiters ist es möglich, mit einem zusätzlichen Popup für jeden Kanal einen Skalierungsfaktor (Zoomwert) vorzugeben.
Wird ein neues Projekt erstellt, weist das GUI jedem Softauge-Kanal automatisch den ersten Variablenwert des Map-Files sowie einen Skalierungsfaktor von eins zu. Für den Fall, dass ein bereits existierendes Projekt geladen wird, übernimmt das GUI für jeden Kanal die zuletzt verwendeten Variablen und Skalierungsfaktoren.
Soll bei einem Softauge-Kanal die Variable oder der Skalierungsfaktor geändert werden, müssen diese anhand der entsprechenden Popups in Bereich 8 des GUI gewählt werden (siehe Abbildung 2.28).

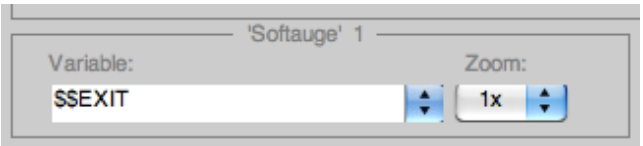


Abbildung 2.29[1]: Kanal 1 des Softauge

Die vier Variablen-Popup der Softauge-Kanäle enthalten eine Auflistung aller Map-File-Variablen. Durch Betätigen der blauen Pfeiltasten können die Popups geöffnet werden. Durch einfaches Auswählen einer Variable in der Liste, wird diese dem entsprechenden Kanal zugewiesen. Mit der Escape-Taste kann das Popup, ohne Änderung der gewählten Variable, wieder geschlossen werden.
Um die Suche nach einer Variable zu vereinfachen, besteht auch hier die Möglichkeit einen Suchstring vorzugeben. Die eingegebene Zeichenfolge wird in diesem Fall unterhalb des Popups dargestellt und die Liste der Map-File-Variablen wird auf jene Variablen eingeschränkt, deren Name mit dem eingegebenen String beginnen.
Da Strukturvariablen im GUI definiert werden, sind diese in der Variablenauflistung des Popups nicht enthalten. Um diese Variablen trotzdem auf einem Softauge-Kanal ausgeben zu können, besteht die Möglichkeit diese mit Hilfe des „Softauge-Buttons“ einem gewünschten Kanal zuzuweisen. Der Softauge-Button befindet sich in Bereich 2 des MATLAB-GUI und ist in jeder Variablenzeile vorhanden (siehe Abschnitt 2.3.5.2).

[1] Quelle: Ausschnitt aus dem MATLAB-GUI

Die Popups für die Zoomwerte enthalten vier vordefinierte Größen, mit denen die normierten Werte der entsprechenden Variablen ein-, zwei-, vier- oder achtfach vergrößert ausgegeben werden können.
Für den Fall, dass die vordefinierten Zoomfaktoren nicht den Anforderungen entsprechen, können diese als normale GUI-Variablen aufgelistet und entsprechend variiert werden.
Kommt es während einer bestehenden Datentransfer-Verbindung zu einer Änderung eines Zoomwertes oder einer Softauge-Variable, sendet das GUI eine entsprechende Aktualisierung an den DSP.
Falls ein Softauge-Kanal einer Strukturvariable entspricht, deren Namen nachträglich geändert wird, aktualisiert das GUI den entsprechenden Namen der Softauge-Variablen. Auch Änderungen der Datentypen von Strukturvariablen, die Einfluss auf bestehende Adressen von Softauge-Variablen haben, werden vom MATLAB-GUI berücksichtigt.

2.3.6 RAM auslesen

Der DSP umfasst zur Speicherung von Daten einen RAM-Speicher. In diesem RAM ist ein Bereich zur Sicherung von Variablenwerten reserviert. Welche Variablen im RAM gesichert werden sollen, wird anhand der DSP-Firmware vorgegeben.
Um die Variablen aus dem RAM auszulesen, muss im GUI der in Abbildung 2.30 gezeigte Button betätigt werden.
Durch die Betätigung dieses Buttons wird eine bestehende Datentransfer-Verbindung unterbrochen und ein Fenster geöffnet. In diesem Fenster ist ein Pfad und ein Name für die Datei anzugeben, in der die gewünschten RAM-Variablen vom GUI gesichert werden sollen. Als Datentyp ist eine „*.mat-Datei“ vorzusehen. Andernfalls kommt es zu einer Fehlermeldung (siehe Abbildung 2.31) und zu einer neuerlichen Aufforderung zur Eingabe von Namen und Pfad für die RAM-Datei.

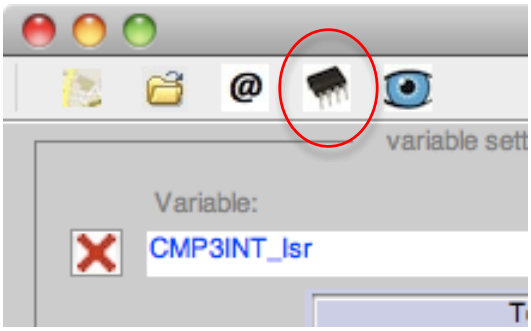


Abbildung 2.30[1]: Button zum Öffnen des RAM-Fenster

[1] Quelle: Ausschnitt aus dem MATLAB-GUI

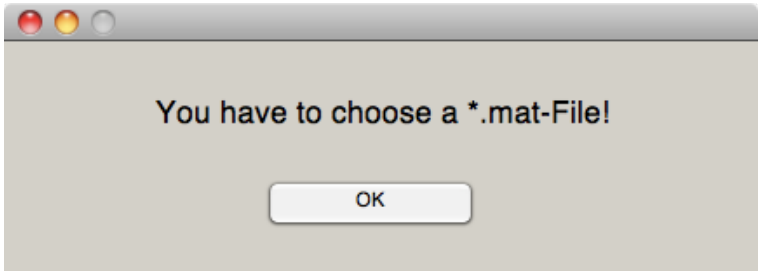


Abbildung 2.31[1]: Fehlermeldung bei falschem Datentyp für RAM-Datei

Wurde eine korrekter Name für die RAM-Datei vorgegeben, öffnet sich das in Abbildung 2.32 gezeigte Fenster.

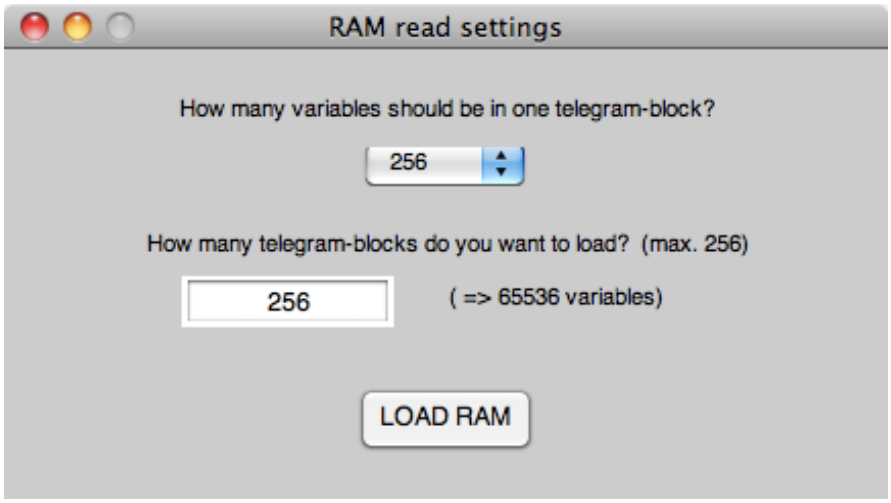


Abbildung 2.32[1]: RAM-Fenster zur Auswahl der Blockgröße und -anzahl

Wie bereits in Abschnitt 2.2.7 erläutert, wird das RAM bei der Initialisierung des DSP in gleich große Blöcke aufgeteilt. Jeder Block enthält dabei eine festgelegte Anzahl an Variablen. Diese Anzahl wird durch die Potenz 2^n bestimmt, wobei der Exponent n eine ganze Zahl zwischen Null und 16 darstellt. Weiters wird vom DSP jedem Block ein Index zugewiesen, wobei der ersten Block im RAM dem niedrigsten Indexwert (Wert Null) entspricht. Alle weiteren Blöcke werden in aufsteigender Reihenfolge indexiert (siehe Abbildung 2.4).
Sollen nun die gesamten RAM-Variablen geladen werden, müssen alle Blöcke vom GUI anhand der Indizes einzeln angefordert und im Anschluss wieder zusammengesetzt werden.
Die Initialisierung des DSP beim Projektstart dient primär zur Überprüfung der Verbindung zum DSP. Aus diesem Grund ist bei diesem Vorgang die Blockgröße nicht einstellbar. Im Gegensatz dazu kommt es beim Laden der RAM-Variablen zu einer Initialisierung, die der Festlegung der Blockgröße dient.

[1] Quelle: MATLAB-GUI

Wie in Abbildung 2.32 zu erkennen, kann die Blockgröße mit Hilfe eines Popups vorgegeben werden. Dabei stehen Größen von 2^4 (= 16) und 2^8 (= 256) Variablen zur Auswahl.
Je größer die Blöcke, desto weniger einzelne Telegramme sind notwendig um den gesamten Inhalt des RAM's auszulesen. Dies hat als Vorteil eine Verkürzung der Ladezeit zur Folge. Im Gegenzug dazu wird jedoch die Byte-Anzahl im Telegramm sehr hoch. Dadurch steigt die Wahrscheinlichkeit, dass ein Telegrammbyte bei einer störanfälligen Verbindung falsch übertragen wird und somit das Telegramm die Prüfsummen-Kontrolle nicht besteht. Dies macht eine neuerliche Anforderung des Telegramms notwendig, wobei aufgrund der Länge des Telegramms erneut ein Telegrammbyte verfälscht werden und dies somit zum Abbruch des Ladevorgangs führen kann.
Wählt man Hingegen eine kleine Blockgröße, sind deutlich mehr Telegramme notwendig, um den gesamten RAM-Inhalt auslesen zu können. Dies führt zu einer längeren Ladedauer der gesamten RAM-Variablen. Kommt es nun aufgrund einer störanfälligen Verbindung zu einer Verfälschung eines Telegrammbytes, so muss auch dieses Telegramm erneut angefordert werden. Da die Telegramme jedoch relativ kurz sind, ist die Wahrscheinlichkeit eher gering, dass das erneut angeforderte Telgramm wieder fehlerhaft ist. Dadurch kann ein Abbruch des Ladevorgangs verhindert werden.
Unterhalb des Popups zur Bestimmung der Blockgröße ist in Abbildung 2.32 ein Eingabefeld zu erkennen. Dieses dient dazu, die gewünschte Anzahl an zu ladenden Blöcken, beginnend mit dem Ersten, festzulegen. Als Blockanzahl ist stets eine ganze positive Zahl vorzusehen! Sind beispielsweise die ersten 100 RAM-Variablen bei einer Blockgröße von 16 Variablen von Interesse, so muss in diesem Eingabefeld eine Blockanzahl von sieben eingetragen werden, wodurch 112 Variablen geladen werden.
Anhand dieser aufgeführten Erläuterungen ist zu erkennen, dass die Wahl der Blockgröße von der Qualität der Verbindung und auch von der gewünschten Anzahl an RAM-Variablen abhängt.
Der Wert in der Klammer neben dem Eingabefeld zeigt an, wie viele Variablen mit der gewählten Blockgröße und -anzahl geladen werden.
Die Angabe der maximal ladbaren Blöcke ist abhängig von der Blockgröße und der Größe des RAM. Damit das GUI in der Lage ist, diesen Wert korrekt darzustellen, muss die Größe des RAM-Speichers in den Software-Parametern gegebenenfalls angepasst werden (siehe Abschnitt 2.3.8). Die Standardeinstellung für diesen Wert beträgt 65536 Variablen zu je zwei Byte.
Wird der Button „LOAD RAM“ betätigt, startet das GUI den zuvor erwähnten Initialisierungsvorgang des DSP um die gewählte Blockgröße zu erstellen. Im Anschluss wird die festgelegte Anzahl an Blöcken geladen und deren Inhalt in der zuvor definierten Datei gespeichert.
Tritt während des Ladevorgangs ein Fehler in der Kommunikation auf, wird dies anhand des in Abbildung 2.33 gezeigten Fensters signalisiert.

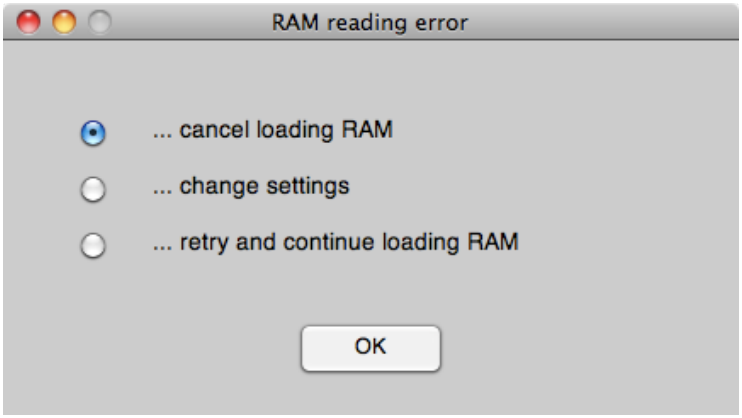


Abbildung 2.33[1]: Meldung bei Fehler während RAM-Ladevorgang

Im Fehlerfall stehen mehrere Optionen zur Verfügung, wie der Fehler gehandhabt werden soll. Es kann der Ladevorgang komplett abgebrochen oder die Blockgröße bzw. Blockanzahl geändert werden. Weiters beseht die Möglichkeit, den Ladevorgang nach dem zuletzt korrekt empfangenen Block fortzusetzen. Dies kann vorteilhaft sein, wenn der Großteil der Blöcke, signalisiert durch einen Fortschrittsbalken, bereits geladen wurde, der letzte Teil jedoch aufgrund einer störanfälligen Verbindung mehrmals falsch übertragen und somit der Ladevorgang vom GUI abgebrochen wurde. Durch die kurze Unterbrechung besteht die Möglichkeit, dass die zuvor betroffenen Blöcke nicht mehr fehlerbehaftet sind und somit die geforderte Anzahl an Blöcken komplett geladen werden kann, ohne den gesamten Ladevorgang neu beginnen zu müssen.

2.3.7 „The Eye“

„The Eye“ ist eine Funktion, die eine grafische Darstellung der empfangenen Variablenwerte über einen bestimmten Zeitraum ermöglicht (siehe Abbildung 2.34).
Die Aktualisierungsrate der darzustellenden Variablenwerte wird dabei durch die Datentransfer-Verbindung vorgegeben und beläuft sich auf maximal fünf bis zehn Werte pro Sekunde. Bei Beachtung des Nyquist-Shannon-Abtasttheorems können mit Hilfe von „The Eye“ somit langsam verlaufende Signale sowie Tendenzen einzelner Parameter dargestellt werden. Ob es sich bei den darzustellenden Signalen um periodische oder nicht periodische Verläufe handelt, ist dabei nicht von Bedeutung.
Nach Ablauf des vorgegebenen Zeitraumes werden die Aufzeichnungen gelöscht, und die Darstellung der Variablenwerte beginnt wieder beim Zeitpunkt t=0.

[1] Quelle: MATLAB-GUI

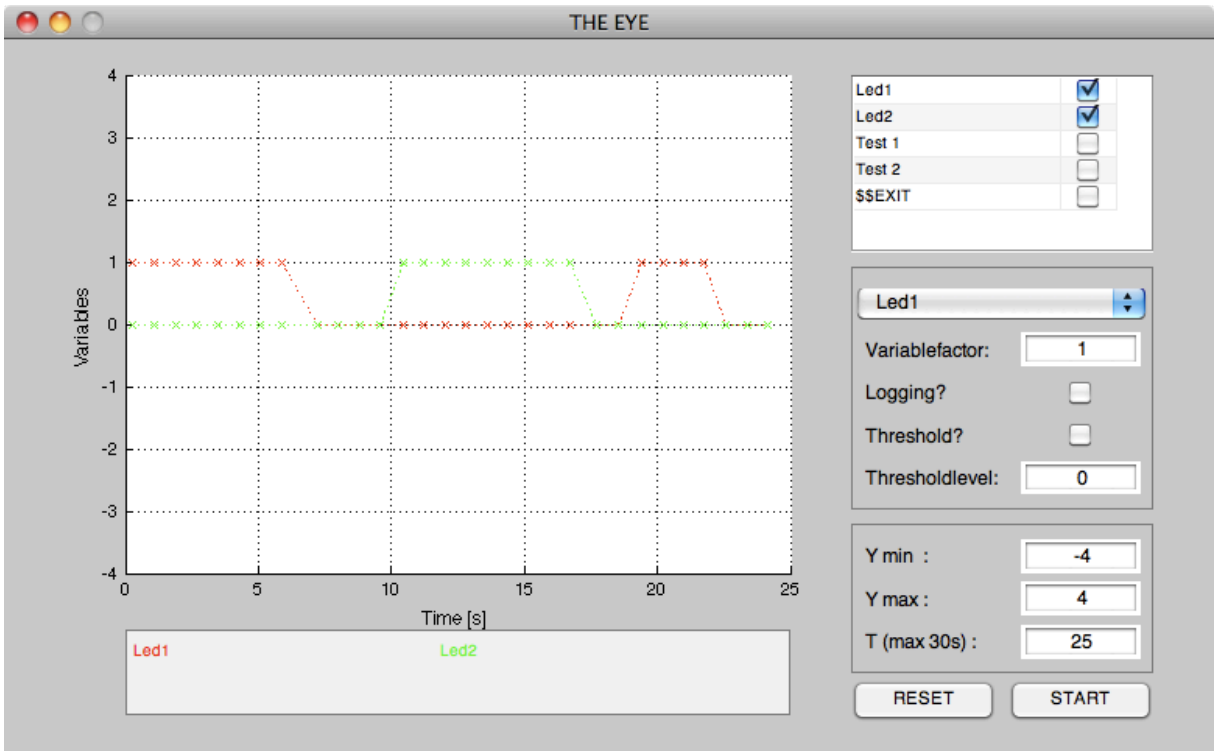


Abbildung 2.34[1]*: Oberfläche der Funktion „The Eye“*
Die Aktualisierungsrate beträgt in diesem Beispiel etwa 1,4 Werte/s.

Das in Abbildung 2.34 gezeigte Fenster erscheint durch Betätigung des in Bild 2.35 markierten Buttons. Ein abermaliges Betätigen dieses Buttons oder ein „Schließen“ des Fensters mit dem entsprechenden Button in der Titelleiste von „The Eye“ schaltet das Fenster wieder in den unsichtbaren Zustand. Alle getätigten Einstellungen und eventuell aktive Aufzeichnungen bleiben erhalten bzw. werden im Hintergrund fortgesetzt.

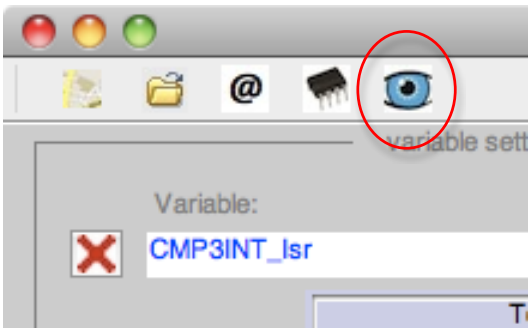


Abbildung 2.35[2]*: Button zum Öffnen von „The Eye“*

[1] Quelle: MATLAB-GUI
[2] Quelle: Ausschnitt aus dem MATLAB-GUI

Die Oberfläche von „The Eye“ teilt sich in zwei Bereiche. Der linke und größere Bereich stellt die Signalverläufe und die zugehörige Legende dar. Der rechte Bereich des Fensters dient den Einstellungen.
Im oberen Teil des Einstellungsbereiches werden die im GUI stets aktuell aufgelisteten Variablen in Tabellenform dargestellt. Ändert sich die Variablenauflistung im GUI, ändert sich auch die „Variablentabelle“ von „The Eye“. Mehrfache Auflistungen von Variablen im GUI werden in der Tabelle nur einfach berücksichtigt.
Mittels der Checkboxen in der Tabelle können bis zu sieben Variablen ausgewählt werden, die von „The Eye“ grafisch dargestellt werden sollen.
Unterhalb der Tabelle besteht die Möglichkeit, die ausgewählten Variablen zu parametrieren. Insgesamt stehen drei Parameter zur Verfügung, die auf jede Variable individuell angepasst werden können. Die anzupassende Variable wird dabei mit Hilfe des Popups ausgewählt.
Der erste Parameter beschreibt den Variablenfaktor. Dieser dient der Multiplikation des normierte Variablenwertes vor dessen Darstellung in „The Eye“. Der Betrag dieses Faktors wird im Feld „ Variablefactor “ vorgegeben.
Der Parameter „Logging“ wird anhand der entsprechen Checkbox aktiviert. Das Logging dient zum Sichern von dargestellten „Eye-Bildern“. Die empfangenen Variablenwerte werden im Diagramm über der Zeit aufgetragen. Ist der maximal darstellbare Zeitpunkt erreicht, wird das Diagramm gelöscht und die Variablenwerte werden wieder bei Null beginnend aufgetragen. Ist bei mindestens einer Variable die Logging-Funktion aktiviert, erstellt „The Eye“ jeweils vor dem Löschen des Diagramms eine „*.mat-Datei“.
Der Suffix „.mat“ bezeichnet einen spezifischen Datentyp von MATLAB. Dateien von diesem Typ enthalten MATLAB-Variablen, die zu einem späteren Zeitpunkt bei Bedarf von MATLAB wieder gelesen werden können.
Im Fall von „The Eye“ enthält die „*.mat-Datei“ eine Zelle, in der alle Variablen aufgelistet sind, deren Logging-Funktion zum Zeitpunkt des Löschens aktiviert waren. Neben dem Variablennamen sind in der Zelle die entsprechend im Bild eingetragenen Werte und Zeitpunkte sowie die jeweiligen Variablenfaktoren enthalten. Als Dateiname für die mat-Datei wird jener Zeitpunkt (Datum und Uhrzeit) verwendet, an dem die Darstellung des Bildes bei Null startete. Anhand dieser Informationen ist es möglich, ein bestimmtes Bild nachträglich zu rekonstruieren. Da die Anzahl der Logging-Dateien, abhängig von der auf der Abszisse aufgetragenen Maximalzeit und des Logging-Zeitraumes, sehr groß werden kann, werden diese in einem eigenen Log-Ordner gesichert. Der Speicherort dieses Ordners ist ident mit dem Speicherort der Projektdatei.
Der dritte Parameter umfasst die Schwellwert-Funktion. Diese wird mittels der entsprechenden Checkbox aktiviert. Der Betrag des Schwellwertes kann unterhalb der Checkbox im Feld „Thresholdlevel“ vorgegeben werden. Ist die Threshold-Funktion aktiviert, wird die entsprechende Variable erst nach Überschreiten des vorgegebenen Schwellwertes in „The Eye“ dargestellt. Kommt es in weiterer Folge zu einem Unterschreiten des Thresholdlevels, hat dies keine Auswirkung mehr auf die Darstellung. Betätigt man bei laufender Darstellung der Variablenwerte den Button „RESET“, wird die Schwellwert-Funktion wieder aktiviert und die Darstellung der Variablen beginnt erst wieder bei Überschreiten des jeweiligen Schwellwertes. Bei Bedarf können die Schwellwerte für jede Variable einzeln vorgegeben und aktiviert werden.

Im unteren Teil des Einstellungsbereiches von „The Eye“ befinden sich die Einstellungen für die Achsenparameter. Dabei können die Minimal- und Maximalwerte der Ordinate (Variablenwerte), sowie die darzustellenden Zeitspanne der Periode auf der Abszisse vorgegeben werden. Als maximale Periode gilt 30 Sekunden. Nach Ablauf dieser Zeit werden die Aufzeichnungen gegebenenfalls in einer Log-Datei gesichert und die Darstellungen der empfangenen Variablenwerte gelöscht. Die folgenden Variablenwerte beginnen in der Illustration wieder bei t=0.
Die beiden Buttons „RESET“ und „START dienen der Steuerung von „The Eye“. Der Button „START“ ändert, je nach Zustand von „The Eye“, seine Aufschrift in „START“ oder „STOP“ und wird in weiterer Folge als Startbutton bezeichnet.
Sollen die durch die entsprechende Checkbox ausgewählten Variablen dargestellt werden, muss der Startbutton betätigt werden. Sofern nun Werte durch eine aktivierte Datentransfer-Verbindung im GUI empfangen werden, kommt es in „The Eye“ zur Darstellung der ausgewählten Variablen. In diesem Zustand kann man die Parameter jederzeit anpassen, wobei die Aufzeichnung bei einer Parameteränderung wieder bei Null startet. Eine Änderung, welche der aufgelisteten Variablen abgebildet werden sollen, ist jedoch nur im gestoppten Zustand von „The Eye“ möglich. Dieser Zustand wird durch abermaliges Betätigen des Startbuttons erreicht. Die bis zu diesem Zeitpunkt aufgezeichneten Daten bleiben in der Grafik erhalten. Erst ein neuerlicher Start löscht die Aufzeichnungen und beginnt mit der Darstellung wieder bei t=0, wobei die Schwellwert-Funktion durch den Neustart wieder aktiviert wird.
Sollte eine Variable aus dem GUI entfernt werden, die zeitgleich in „The Eye“ dargestellt wird, so wird diese auch aus der Grafik entfernt.
Mit Hilfe des Resetbuttons ist es möglich, die Darstellung der Variablenwerte vor Ablauf der vorgegebenen Periodendauer wieder bei Null zu beginnen. Außerdem wird, wie bereits erläutert, durch Betätigung des Resetbuttons gegebenenfalls die Schwellwertfunktion wieder aktiviert. Eine aktivierte Schwellwertfunktion bedeutet, dass eine Variable erst nach Überschreiten des vorgegebenen Schwellwertes dargestellt wird. Ist dies der Fall, kommt es zu einer Deaktivierung der jeweiligen Schwellwertfunktion und die folgenden Variablenwerte werden alle abgebildet.
Sollte das Fenster von „The Eye“ nicht sichtbar sein, kann auch im GUI dessen Zustand erkannt werden. Ein blaues Auge (siehe Abbildung 2.35) symbolisiert, dass sich „The Eye“ im deaktivierten Zustand befindet und keine Aufzeichnung stattfindet. Ist das Auge rot gefärbt, liegt ein aktiver Zustand von „The Eye“ vor. Dieses zeichnet somit die Daten der gewählten Variablen entsprechend ihrer Parameter auf, sobald bzw. solange eine aktive Datentransfer-Verbindung besteht.
Sollte die Datentransfer-Verbindung noch nicht hergestellt, „The Eye“ aber über den entsprechenden Button bereits aktiviert sein, nimmt dieses einen „Bereitschafts-zustand“ ein. Ist nun zu einem späteren Zeitpunkt die Datentransfer-Verbindung aufgebaut, so wird ein Reset von „The Eye“ durchgeführt und die grafische Darstellung der gewählten Variablen begonnen. Eventuell vor dem Start der Verbindung vorhandene Aufzeichnungen werden durch den Reset aus „The Eye“ entfernt!

2.3.8 Software-Parameter

```
458
459
460     %------------- allgemeine Software-Parameter -------------%
461 -   GUI_handles.maxvar = 24;           % maximale Variablenanzahl
462 -   GUI_handles.norm = 4096;           % Normierungswert 2^12
463 -   GUI_handles.long_default = 0;      % Defaultwert fuer long/int Auswahl
464 -   GUI_handles.no_change = 0;         % diese Variable signalisiert, ob das
465                                        % letzte Projekt automatisch geladen
466                                        % werden soll, ohne danach Aenderungen
467                                        % im GUI vornehmen zu koennen (Wert 1)
468 -   GUI_handles.debugfolder = 'Debug';% Standardname des Ordners, in dem das
469                                        % Mapfile beim Flashvorgang gespeichert
470                                        % wird und bei jedem Flashvorgang
471                                        % ueberschrieben wird
472 -   GUI_handles.auto_projectload = 1; % diese Variable markiert, ob zum
473                                        % Projektstart der Projektname abhaengig
474                                        % vom Map-File-Namen automatisch
475                                        % generiert bzw. geladen werden soll
476                                        % (Wert 1) oder durch ein Eingabefenster
477                                        % vom Benutzer erstellt/geladen werden
478                                        % soll (Wert 0)
479 -   GUI_handles.RAMsize = 65536;       % Variablen im RAM zu je 2 Byte
480     %------------- allgemeine Software-Parameter -------------%
481
```

Abbildung 2.36[1]: Block der allgemeinen Software-Parameter

```
543
544     %------------- Verbindungs-Parameter -------------%
545     % Paramter fuer  TCP und serielle Verbindung:
546 -   GUI_handles.connection.Buffersize = 1024;          % Buffergroesze der
547                                                       % Verbindung
548 -   GUI_handles.connection.RAMProtokolllaenge = 2^8; % Laenge des RAM-Blocks
549                                                       % beim Verbindungscheck;
550                                                       % muss immer eine Zahl 2^n
551                                                       % sein, wobei n<=16 gilt!
552 -   GUI_handles.connection.Checksum_timeout = 8;      % Zeit, nach der das
553                                                       % Empfangen der DSP-
554                                                       % Antwort abgebrochen wird
555                                                       % (wegen falscher
556                                                       % Checksumme, keiner
557                                                       % Antwort,...)
558 -   GUI_handles.connection.con_timout = 2;            % Timeout der Verbindung
559                                                       % (solange wird vom Befehl
560                                                       % 'fread' auf Antwort
561                                                       % gewartet)
562
563     % Paramter fuer serielle Verbindung:
564 -   GUI_handles.connection.Baud = 115200;             % Baudrate, die bei
565                                                       % serieller Verbindungzu
566                                                       % Beginn in der
567                                                       % Auswahlliste angezeigt
568                                                       % wird
569 -   GUI_handles.connection.databits = 8;
570 -   GUI_handles.connection.parity = 'none';
571 -   GUI_handles.connection.StobBits = 1;
572 -   GUI_handles.connection.FlowControl = 'none';
573 -   GUI_handles.connection.Baudliste = ...
574         {'115200','57600','38400','19200','9600'}';  % moegliche Baudraten
575     %------------- Verbindungs-Parameter -------------%
576
```

Abbildung 2.37[1]: Block der Verbindungs-Parameter

[1] Quelle: Ausschnitt aus der MATLAB-Funktion „IEAM_Com.m“

Anhand der Software-Parameter besteht die Möglichkeit, das MATLAB-GUI an die Anforderungen anzupassen. Prinzipiell wird dabei zwischen allgemeinen Software-Parametern und Verbindungs-Parametern unterschieden. Beide Parametertypen befinden sich in der MATLAB-Funktion „IEAM_Com.m“ und sind entsprechend zusammengefasst bzw. markiert. Die Abbildungen 2.36 und 2.37 zeigen die beiden Blöcke innerhalb dieser Funktion.
Alle Parameter werden in einer Handle-Struktur gesichert und stehen somit bei Bedarf in jeder anderen Funktion des GUI zur Verfügung.

2.3.8.1 Allgemeine Software-Parameter

Insgesamt stehen bei den allgemeinen Software-Parametern sieben Variablen zur Verfügung, um das MATLAB-GUI entsprechend den geänderten Anforderungen anpassen zu können. Diese Variablen werden in Funktion und Wertebereich in Tabelle 2.4 dargestellt.

Strukturvariable	Funktion	Wertebereich
maxvar	Mit dieser Variable wird die maximal erlaubte Anzahl an GUI-Variablen festgelegt.	Default: 24 Bereich: 1 bis 24 Nur ganze Zahlen!
norm	Dieser Parameter definiert den Normierungswert der Variablen im GUI.	Default: 2^{12} = 4096
long_default	Der Wert dieser Variable signalisiert, welcher Datentyp als Standardtyp verwendet werden soll. Der Standardtyp wird z.B. herangezogen, wenn eine Datentyp-Abfrage mit der Eingabe von „Enter“ bestätigt wird.	Default: 0 Möglichkeiten: 0 oder 1 0 ... Datentyp Integer 1 ... Datentyp Long
no_change	Der Wert dieses Parameters signalisiert, ob das GUI nach dem Start eines Projekts in den Funktionen eingeschränkt wird oder alle Funktionen zur Verfügung stehen sollen. Eine Einschränkung kann erfolgen, wenn die Software an Dritte weitergegeben wird, diese jedoch nur einen Teil der Möglichkeiten nutzen dürfen. Die Festlegung, welche GUI-Elemente nicht zur Verfügung stehen sollen, wird in Abschnitt 2.3.8.3 erläutert.	Default: 0 Möglichkeiten: 0 oder 1 0 ... keine Einschränkung 1 ... aktivierte Einschränkung
debugfolder	Das beim Kompilieren erzeugte Map-File wird in einem Ordner abgelegt, der ebenfalls beim Flashvorgang erzeugt wird. Der bisherige Ordner inklusive seinem Inhalt wird dabei durch den neuen Ordner überschrieben. Aus diesem Grund darf die Projektdatei nicht den selben Pfad wie das Map-File aufweisen, wenn sich dieses in besagtem Ordner befindet. Anhand der Strukturvariable „debugfolder“ wird dem GUI mitgeteilt, welchen Namen dieser sich stets erneuernde Ordner trägt.	Default: 'Debug’

auto_projectload	Bei einem Projektstart muss ein Map-File ausgewählt werden. Anhand dieses Map-Files wird automatisch eine Projektdatei und ein passender Speicherpfad generiert. Will man den Namen der Projektdatei und den Pfad manuell erstellen, kann dies mit Hilfe dieser Strukurvariable signalisiert werden.	Default: 0 Möglichkeiten: 0 oder 1 0 ... automatisches erstellen des Projekts 1 ... manuelles erstellen des Projekts
RAMsize	Anhand dieses Parameters wird dem GUI mitgeteilt, wie viele Variablen im RAM des DSP sein können. Der angegebene Wert entspricht dabei nicht der Anzahl an Byte, sondern der tatsächlichen Anzahl an Variablen! (Eine RAM-Variable ist stets vom Datentyp Integer und benötigt somit zwei Byte.)	Default: 65536 Möglichkeiten: 2^n n entspricht dabei einer ganzen Zahl größer 0!

Tabelle 2.4: Übersicht über die allgemeinen Software-Parameter

Alle aufgelisteten Variablen sind der Struktur „GUI_handles“ zugewiesen (GUI_handles.Variable). Um in der Tabelle jedoch eine besseres Übersicht zu wahren, werden diese hier ohne dem Strukturnamen dargestellt.

2.3.8.2 Verbindungs-Parameter

Zur Anpassung der Verbindungs-Parameter stehen zehn Variablen zur Verfügung, die in Tabelle 2.5 näher erläutert werden.

Variable	Funktion	Wert
Buffersize	Der Wert dieser Variable bestimmt die von MATLAB zu verwendende Eingangsbuffergröße der Verbindung in Byte.	Default: 1024
RAMProtokolllaenge	Dieser Parameter gibt an, wie groß die Blockgröße in Byte bei der ersten Initialisierung des DSP sein soll. Diese Initialisierung dient nur zur Kontrolle der Verbindung zwischen DSP & GUI und ist nicht entscheidend für das Auslesen der RAM-Variablen. Die Blockgröße für das Auslesen wird im entsprechenden Menü (siehe Abschnitt 2.3.6) vorgegeben.	Default: 2^8 Möglichkeiten: 2^n n ist dabei eine natürliche Zahl, wobei 2^n maximal der Anzahl an Variablen im RAM entsprechen darf!
Checksum_timeout	Der Wert dieser Variable gibt die Zeit in Sekunden an, innerhalb der ein Telegramm korrekt empfangen werden muss. Sollte in dieser Zeit, trotz mehrmaliger Anforderung der DSP-Antwort, die Prüfsummen-Prüfung (siehe Abschnitt 2.2.6) nicht bestanden werden bzw. keine Daten empfangen werden, wird die bestehende Verbindung des GUI beendet und eine Fehlermeldung ausgegeben.	Default: 8

con_timout	Sendet das GUI ein Telegramm an den DSP, wird von diesem stets eine Antwort erwartet. Sollte diese Antwort das GUI nicht innerhalb eines definierten Zeitraumes erreichen oder diese die Prüfsummen-Prüfung nicht bestehen, sendet das GUI das Telegramm erneut an den DSP. Der Wert des Parameters „con_timout“ legt dabei den definierten Zeitraum in Sekunden fest, innerhalb dessen eine Antwort erwartet wird.	Default: 2
Baud	Der Wert dieser Variable dient als Defaultwert für die Baudrate. Dieser wird im entsprechenden Popup dargestellt, wenn das Fenster zur Einstellung der Verbindung im Zuge eines Projekts zum ersten Mal geöffnet wird. Sollte eine andere Baudrate ausgewählt werden, wird diese gespeichert und anstelle des Defaultwerts beim nächsten Öffnen des Fensters im Popup angezeigt.	Default: 115200 Möglichkeiten: Alle in der Variable „Baudliste“ definierten Werte können als Defaultwert verwendet werden.
databits	Dieser Parameter repräsentiert die Anzahl an Datenbits der seriellen Verbindung.	Default: 8
parity	Anhand dieser Variable wird die Paritätseinstellung der seriellen Verbindung festgelegt.	Default: 'none’
StopBits	Der Wert dieser Variable definiert die Anzahl an Stoppbits der seriellen Verbindung.	Default: 1
FlowControl	Mit Hilfe dieses Parameter lässt sich die Datenflusssteuerung der seriellen Verbindung festlegen	Default: 'none’
Baudliste	Die Variable „Baudliste“ enthält ein Liste von auswählbaren Baudraten. Diese Baudraten werden im entsprechenden Popup des Fensters zur Verbindungseinstellung aufgelistet. Im Programm wird die Liste als „Cell Array“ gespeichert.	Default: {'115200', '57600', '38400', '19200', '9600'}'

Tabelle 2.5: Übersicht über die Verbindungs-Parameter

Alle aufgelisteten Variablen sind der Struktur „GUI_handles.connection“ zugewiesen. Um in der Tabelle jedoch eine besseres Übersicht zu wahren, werden diese hier ohne dem vollständigen Strukturnamen (GUI_handles.connection.Variable) dargestellt.

2.3.8.3 Einschränkung des Funktionsumfangs

Wie bereits in Tabelle 2.4 bei der Variable „GUI_handles.no_change“ erläutert, besteht die Möglichkeit, den Funktionsumfang des MATLAB-GUI einzuschränken. Dies ist insofern von Interesse, als dass dadurch eine Nutzung der Software außerhalb des Projektrahmens von Projektpartnern oder Außenstehenden verhindert werden kann.
Um eine Einschränkung vornehmen zu können, sind folgende drei Anforderungen zu erfüllen:

- Zuerst muss ein Projekt angelegt werden, das alle Variablen und Gruppen enthält, die zur Verfügung stehen sollen. Wird das angelegte Projekt geschlossen, erzeugt das GUI eine Projektdatei mit allen relevanten Projektdaten. Der Speicherort dieser Datei orientiert sich dabei am Speicherort des Map-Files (siehe Abschnitt 2.3.2).
- Danach muss man die Variable „GUI_handles.no_change“ in der Funktion „IEAM_Com.m“ mit dem Wert 1 versehen. Dadurch werden beim nächsten Start des MATLAB-GUI alle dafür vorgesehenen Elemente deaktiviert.
- Zum Abschluss muss die Software des MATLAB-GUI vor Manipulationen geschützt werden. Dies geschieht mit der von MATLAB für diesen Zweck zur Verfügung gestellten Möglichkeit, Funktionen der Software (m-Files) in nicht lesbaren „p-Code“ (p-Files) umzuwandeln. (siehe Abschnitt 2.3.9).

Ist die Einschränkung des Funktionsumfangs aktiviert, bleiben jedenfalls folgende GUI-Funktionen erhalten:

- Die Zoomwerte der Softauge-Kanäle können bestimmt werden.
- Die Verbindungseinstellungen können vorgenommen werden.
- Eine eventuell vorhandene Gruppe kann ausgewählt werden.
- Ein Map-File bzw. ein Projekt kann bestimmt werden.
- Die Datentransfer-Verbindung kann gestartet werden.
- Die Aktualisierungsrate kann variiert werden.

Weiters können zusätzlich, je nach Bedarf, einzelne GUI-Funktionen bzw. GUI-Elemente aktiviert werden.
Alle deaktivierten Elemente sind am Ende der Softwarefunktion „makelist.m“ in einem Block aufgelistet (siehe Abbildung 2.38). Diese Elemente kann man einzeln aktivieren, indem man vor dem Erstellen des p-Codes die entsprechenden Code-Zeilen in der Funktion „makelist.m“ entfernt bzw. als Kommentar markiert („ausklammert“).
Soll beispielsweise der Button von „The Eye“ in der Menüleiste des GUI aktiviert und somit die Verwendung von „The Eye“ ermöglicht werden, so muss vor die Zeile:

```
set(GUI_handles.eye,'Enable','off');
```

das Zeichen „%“ gesetzt werden:

```
%set(GUI_handles.eye,'Enable','off');
```

Dadurch wird die Code-Zeile als Kommentar betrachtet und somit nicht ausgeführt.

Um das eingeschränkte MATLAB-GUI letztlich nutzen zu können, ist der Softwareordner der geschützten MATLAB-Software (p-Files), die gewünschte

Projektdatei und das zugehörige Map-File notwendig. Dabei ist zu beachten, dass das Map-File und die Projektdatei am selben Ort gespeichert werden müssen, sofern die Projektdatei beim Start nicht manuell ausgewählt wird (siehe Tabelle 2.4, Variable „auto_projectload“).
Der Start des MATLAB-GUI erfolgt, wie in Abschnitt 2.3.2 beschrieben, durch das Ausführen der Funktion „IEAM_Com.p“.
Sollte es notwendig sein, die Variablenauflistung nachträglich zu ändern, muss nur die Projektdatei erneuert werden. Die neue Projektdatei kann dabei von einem anderen GUI erstellt werden, wobei diese aber auf dem selben Map-File beruhen muss wie die bisherige Projektdatei! Andernfalls ist auch das Map-File zu aktualisieren! Dem Projektpartner kann somit durch das Übermitteln einer neuen Projektdatei und gegebenenfalls eines neuen Map-Files das MATLAB-GUI einfach an den aktuellen Projektstatus angepasst werden.

```
596
597       % Wenn die Variable 'GUI_handles.no_change' mit dem Wert 1 versehen wird,
598       % werden beim Ausfuehren des MATLAB-GUI die unten aufgefuehrten Elemente
599       % im GUI deaktiviert! Dadurch ist eine Einschraenkung des Funktionsumfangs
600       % des MATLAB-GUI moeglich.
601
602       % Sollte ein Element doch aktiviert werden, muss man die entsprechende
603       % Zeile im folgenden Abschnitt zu Beginn mit "%" versehen!
604 -     if GUI_handles.no_change
605
606           % Toolbar
607 -         set(GUI_handles.loadtool,'Enable','off');
608 -         set(GUI_handles.ram,'Enable','off');
609 -         set(GUI_handles.eye,'Enable','off');
610
611           % Gruppe-Buttons
612 -         set(GUI_handles.addgroup,'Enable','off');
613 -         set(GUI_handles.editgroup,'Enable','off');
614
615 -         for index = 1:1:GUI_handles.maxvar
616
617               % Add-/Remove-Button
618 -             set(GUI_handles.variablen(index).remove,'Enable','off');
619 -             set(GUI_handles.variablen(index).add,'Enable','off');
620
621               % Variablen-Popup
622 -             set(GUI_handles.variablen(index).popup_but,'Enable','off');
623
624               % Strukturname veraenderbar
625 -             set(GUI_handles.variablen(index).edit,'Enable','off');
626
627               % Struktur-Popup
628 -             set(GUI_handles.variablen(index).struktur,'Enable','off');
629
630               % Long-/Integer-Abfrage
631 -             set(GUI_handles.variablen(index).long_int,'Enable','off');
632
633 -         end
634
635           % Softauge-Variablen => Popupliste deaktivieren
636 -         for sf = 1:1:4
637
638 -             set(GUI_handles.softauge(sf).popup_but,'Enable','off');
639
640 -         end
641
642 -     end
643
```

Abbildung 2.38[1]: Block zur Anpassung der Funktionseinschränkung

[1] Quelle: Ausschnitt aus der MATLAB-Funktion „makelist.m“

2.3.9 Schützen der Software mit Hilfe von „p-Code“

Um eine Softwarefunktion vor Manipulationen schützen zu können, bietet MATALB die Möglichkeit, Funktionen in „p-Code“ („Protected-Code“) umzuformen. Eine p-Funktion kann wie gewohnt ausgeführt werden, ist jedoch in einem Editor nicht lesbar. Eine Änderung des Codes ist somit nicht möglich!
Sollen alle Funktionen der Software des MATLAB-GUI geschützt werden, sind dafür vier Anforderungen zu erfüllen:

- Zuerst muss ein Ordner erstellt werden, in dem die geschützten Dateien abgelegt werden sollen.
- Danach muss dieser Ordner in MATLAB als „Current Directory“ gewählt werden.
- Im Anschluss ist im „Command Window“ von MATLAB der Befehl

  pcode('Pfad des ungeschützten Softwareordners')

  einzugeben und mit „Enter“ zu bestätigen. Der Ausdruck „Pfad des ungeschützten Softwareordners“ beschreibt dabei jenen Ordner inklusive seinem Pfad, in dem alle zu schützenden Software-Funktionen („m-Files“) abgelegt sind.

  Bespiel für Mac: pcode('Users/Name/Desktop/Softwareordner')

  Beispiel für PC: pcode('C:\Dokumente und Einstellungen\
  Name\Desktop\Softwareordner')

- Um schließlich die geschützte Software ausführen zu können, muss noch eine Kopie der Datei „icons.mat“ aus dem Softwareordner dem eben erstellten Ordner hinzugefügt werden! Ohne dieser Datei kann die Funktion „IEAM_Com.p“ nicht ausgeführt werden, da alle im GUI verwendeten Symbole aus dieser mat-Datei geladen werden.

Das Starten bzw. Anlegen eines Projekts erfolgt mit der geschützten Software in gleicher Weise, wie in Abschnitt 2.3.2 beschrieben!

3. Aufbau der MATLAB-Software

3.1 Allgemein

Die gesamte Software des MATLAB-GUI ist in mehrere Funktionen aufgeteilt, denen eine spezifische Aufgabe im Ablauf des Programms zugeteilt ist.
Der Vorteil der Aufgabenaufteilung in unterschiedliche Funktionen liegt darin, dass die Software dadurch eine bessere Übersichtlichkeit und Wartbarkeit erhält.
Die folgenden Abschnitte sollen nun die wichtigsten Funktionen in einer kurzen Übersicht aufzeigen, um eine Erweiterung bzw. Wartbarkeit des MATLAB-GUI zu ermöglichen. Für nähere Details zu den Codezeilen sei auf die Kommentare in den Funktionen verwiesen.
Manche Funktionen enthalten Grafikobjekte, deren „Properties“ je nach Betriebssystem unterschiedlich sind. Der Grund dafür liegt darin, dass MATLAB auf einem MAC-System den Bezugspunkt für die Position und die Größe des Grafikobjekts fallweise anders wählt als auf einem Windows-System.
Weiters sind alle Positions- und Größenangaben in den Grafikobjekten normiert, um ein „Resize“ der entsprechenden „Figure“, in der die Objekte enthalte sind, zu ermöglichen.
Im Aufbau der Software wird auf eine Verwendung von Kontextmenüs verzichtet. MATLAB hat die Eigenart, Kontextmenüs nur dann zu öffnen, wenn das Grafikobjekt, auf das sich das Kontextmenü bezieht, deaktiviert ist.
Auf die Verwendung von globalen Variablen wird verzichtet. Stattdessen werden alle relevanten Daten in einem „Figure-Handle“ in Form einer Struktur gesichert. Dieses Handle steht bei Bedarf in allen Funktionen zur Verfügung, solange die Figure existiert. Da die Figure des MATLAB-GUI vom Start bis zum Beenden des GUI’s besteht, bietet sich deren Handle zur Sicherung aller relevanten Daten an.

3.2 GUI

3.2.1 Funktion IEAM_Com.m

Das MATLAB-GUI wird durch die Funktion „IEAM_Com“ gestartet. Diese Funktion erstellt dabei eine Figure mit allen grafischen Objekten des GUI, die je nach Bedarf sichtbar werden. Das Erzeugen aller Grafikobjekte beim Programmstart erhöht die Performance des Programmablaufs, da das Ändern der Sichtbarkeit eines Objekts weniger Ressourcen benötigt als das Erstellen.
Diese erstellte Figure stellt das MATLAB-GUI dar, dessen Handle zur Speicherung der Daten herangezogen wird. In diesem Handle werden neben den Variablendaten auch die Kennung (Identifier) von allen notwendigen Grafikobjekten in Strukturform gesichert.
In der Funktion „IEAM_Com“ werden, neben der Erstellung aller grafischen Elemente, die gesamten Software-Parameter (siehe Abschnitt 2.3.8) vorgegeben.
Weiters wird die Struktur für die Gruppenverwaltung angelegt, die Menüleiste („Toolbar“) erstellt und, falls vorhanden, die IEAM_Com_ini.mat eingelesen. Diese Datei enthält Informationen (Name und Pfad) über das zuletzt verwendete Projekt.
Mit Hilfe zweier for-Schleifen werden die Variablenzeilen inklusive einer Variablen-struktur für die jeweiligen Variablendaten sowie die Softauge-Kanäle erstellt.

Die Variablenstruktur enthält für jede Variablenzeile Informationen über:

- den Namen,
- die Adresse,
- eine eventuell vorhanden Variablenstruktur,
- den zuletzt geschriebenen und empfangenen Wert,
- den Datentyp,
- die Gültigkeit der Variable sowie
- ob der im GUI eingetragene Wert bereits gesendet wurde.

Die Popups der Variablenzeilen und der Softauge-Kanäle entsprechen nicht den von MATLAB zur Verfügung gestellten Standard-Popups, sondern sind aus einem Textfeld (popup_sel), einem „Pushbutton“ (popup_but) und einer „Listbox“ (popup) zusammengesetzt.
Der Grund dafür liegt darin, dass innerhalb dieser Popups die Variablensuche erleichtert werden soll. Tippt man jedoch in einem Standard-Popup einen Buchstaben ein, wird stets jene Zeile markiert, deren Eintrag mit diesem Buchstaben beginnt. Es wäre somit nicht möglich, einen kompletten Suchstring vorzugeben.
Mit Hilfe der Aufteilung des Popups in einzelne Elemente kann jedoch für den Pushbutton und die Listbox eine Funktion („variablenfokus.m“ bzw. „softaugefokus.m“) erstellt werden, die bei Eingabe eines Zeichens ausgeführt wird (Property „KeyPressFcn“). Anhand dieser Funktion wird der eingegebene String gesichert und die Variablenliste in der Listbox (popup) auf jene Variablen eingeschränkt, deren Namen mit dem String beginnen.
Die Listbox für jede Variablenzeile bzw. jeden Softauge-Kanal wird am Ende der Funktion „IEAM_Com.m“ in eigenen for-Schleifen erstellt. Der Grund dafür liegt darin, dass Grafikobjekte von MATLAB überlagert dargestellt werden, wobei das zuletzt verwendete Objekt oben auf liegt. Diese Darstellung der Listen über allen anderen Objekten ist für das selbst erstellte „Popup“ erwünscht.
Das Grafikobjekt „edit“, hierbei handelt es sich um ein editierbares Textfeld, dient der Darstellung des Namens eines Strukturelements und wird nur in diesem Fall sichtbar. Im Gegenzug werden die drei Elemente des Popups (Button, Textfeld und Listbox) unsichtbar.
Für die Auswahl eines Softauge-Kanals in einer Variablenzeile ist ein Button (softauge) vorgesehen. Bei Betätigung dieses Buttons erscheint eine „Listbox“ (softaugewindow), die nach einem bestimmten Zeitraum wieder unsichtbar wird.
Ebenfalls eine Listbox (editwindow) erscheint bei Betätigung des Buttons „editgroup“. Auch diese wird nach einem vorgegebenen Zeitraum wieder unsichtbar.
Anhand der Timer „edit_timer“ und „softauge_timer“ wird der Zeitraum festgelegt, nach dem diese Listen wieder unsichtbar werden. Der Timer „name_edit“ gibt den Zeitraum vor, innerhalb dessen der Name einer Gruppe geändert werden kann.

3.2.2 Funktion makelist.m

Wird eine Variable entfernt oder hinzugefügt, eine andere Gruppe augewählt oder der Verbindungsstatus geändert, so muss die Oberfläche des GUI geändert werden. Diese Änderung wird stets mit Hilfe der Funktion „makelist.m“ durchgeführt. Diese ist verantwortlich dafür, dass betreffende Objekte bei Bedarf aktiviert bzw. sichtbar werden. Eine Ausnahme bildet dabei das Empfangen und Senden von Daten. Hierbei werden die Werte direkt in die entsprechenden Felder eingetragen, um den Zeitbedarf gering zu halten.
Die Funktion „makelist.m“ wird stets innerhalb anderer Funktionen anhand der Codezeile makelist(Handle) aufgerufen. Der Ausdruck „Handle“ repräsentiert dabei das Handle der GUI-Figure, welches als Parameter an die Funktion übergeben wird.
Der Aufbau der Funktion ist in mehrere Bereiche gegliedert:

- Verbindungsstatus im GUI aktualisieren
- Alle Objekte der Variablenzeilen unsichtbar setzen und die Objekte in Standardfarben darstellen
- Variablenzeilen durch eine for-Schleife aktualisieren
- Softauge-Variablen aktualisieren
- Tools im GUI aktivieren bzw. deaktivieren, je nach dem ob ungültige Variablen im GUI vorhanden sind
- Gruppenliste aktualisieren
- Variablenliste in „The Eye“ aktualisieren
- Block zur Auswahl der Elemente, die im eingeschränkten Funktions-zustand nicht aktiviert werden sollen

Durch die beiden Funktionen „IEAM_Com.m“ und „makelist.m“ wird die gesamte Benutzeroberfläche des MATLAB-GUI erstellt und verwaltet.

3.3 Start und Speichern eines Projekts

3.3.1 Funktion startgui.m

Die Funktion „startgui.m“ wird in der Funktion “IEAM_Com.m“ und als Callback bei Betätigung des „Maptool-Buttons“ ausgeführt.
Als Übergabeparameter wird die GUI-Kennung (GUI-Identifier) der GUI-Figure benötigt, anhand derer in der Funktion „startgui.m“ mit dem Befehl „guidata“ das Handle geladen werden kann.
Da Callbackfunktionen von MATLAB automatisch zwei weitere Übergabeparameter erhalten, müssen diese beim manuellen Aufruf der Funktion als Dummyvariablen hinzugefügt werden. Aus diesem Grund erfolgt der Aufruf in der Funktion „IEAM_Com.m“ mittels der Codezeile startgui(0,0,GUI). Die beiden ersten Parameter mit dem Wert 0 entsprechen den Dummyvariablen und können jeden Wert annehmen. Der Parameter GUI repräsentiert die GUI-Kennung der GUI-Figure.
Wird die Funktion „startgui.m“ ausgeführt, bildet diese ein Start-GUI, in dem die Auswahl zum Beenden des Programms oder zum Laden des Map-Files besteht. Mit dem Pushbutton wird die Auswahl bestätigt und die Callbackfunktion „startbutton.m“ ausgeführt.

3.3.2 Funktion startbutton.m

„startbutton.m“ stellt die Callbackfunktion des Buttons im Start-GUI dar (siehe Abschnitt 3.3.1). Diese Funktion schließt, abhängig von der Auswahl im Start-GUI, das MATLAB-GUI oder öffnet bzw. legt ein neues Projekt an.
Ist die Option zum Öffnen eines neuen Projekts gewählt worden, werden von der Funktion „startbutton.m“ mehrere Aufgaben wahrgenommen:

- Ist bereits ein Projekt im GUI geladen, wird dieses in der Projektdatei gespeichert und der Ursprungszustand im GUI hergestellt.
- Danach wird mit Hilfe der Funktion „map2container.m“ (siehe Abschnitt 3.4.1) das neue Map-File eingelesen.
- In weiterer Folge wird unterschieden, ob die Projektdatei manuell gewählt oder automatisch erstellt werden soll.

- Soll die Projektdatei **manuell** erstellt werden, folgt eine Aufforderung zur Eingabe des Projektpfad und -namen.
- Existiert die gewählte Projektdatei bereits, wird diese geladen und alle enthaltenen GUI-Daten (GUI-Variablen, Gruppen, Softauge-Variablen,...) werden entsprechend in das Handle der GUI-Figure übertragen.
- Im nächsten Schritt wird der Titel des GUI aktualisiert sowie der Ordner für die Log-Files von „The Eye“ erstellt.

- Ist eine **automatische** Erstellung der Projektdatei vorgesehen (siehe Tabelle 2.4, Variable „auto_projectload“), wird diese aus dem Pfad und dem Namen des Map-File generiert.
- Existiert bereits eine Datei, deren Namen und Pfad identisch ist mit der automatisch Erstellten, wird diese geladen. Alle in dieser Datei enthaltenen GUI-Daten werden in das Handle der GUI-Figure übertragen.
- Im Anschluss kommt es ebenfalls zu einer Aktualisierung des GUI-Titels und der Erstellung des Ordners für die Log-Files von „The Eye“.

3.3.3 Funktion figure_close.m

Die Funktion „figure_close.m“ wird ausgeführt, wenn das MATLAB-GUI bzw. die GUI-Figure geschlossen wird.
Sollte die Datentransfer-Verbindung aktiv sein, gibt diese Funktion eine Meldung aus, dass die Verbindung zuerst beendet werden muss. Andernfalls werden die erstellte Verbindung und alle verwendeten Timer gelöscht und alle relevanten GUI-Daten automatisch in der Projektdatei gespeichert. Der Pfad der Projektdatei entspricht dabei, je nach Einstellung, dem manuell oder automatisch erstellten Speicherort (siehe Abschnitt 3.3.2).

Als relevante GUI-Daten werden gespeichert:

- die Anzahl an aufgelisteten Variablenzeilen
- die verwendete IP-Adresse
- der verwendete Port
- die Gruppenstruktur
- die dargestellte Gruppenliste

- die aktuelle Gruppennummer
- den verwendeten Com-Port
- die eingestellte Baudrate
- die verwendete Verbindungsart
- die relevanten Strukturelemente des Softauge
- die relevanten Strukturelemente der GUI-Variablen

3.4 Map-File laden und Variablen kontrollieren

3.4.1 Funktion map2container.m

Anhand der Funktion „map2container.m“ wird vom MATLAB-GUI das Map-File eingelesen und die Variablen-Adressen-Zuweisung in einem Container gespeichert.
Diese Funktion wird im Callback „startbutton.m“ und der Funktion „RXon.m“ mit der Codezeile map2container(0, 0, 0 oder 1, GUI-Identifier) aufgerufen.
Die beiden ersten Übergabeparameter mit dem Wert 0 stellen zwei Dummyvariablen dar, die als Reserve für weitere mögliche Anwendungen dienen. Der dritte Parameter signalisiert der Funktion, ob eine Aufforderung zur Eingabe eines neuen Pfads zum Map-File erfolgen soll (Wert 1). Der vierte Parameter stellt die Kennung der GUI-Figure dar, mit deren Hilfe das GUI-Handle geladen werden kann.
Die Funkion „map2container.m“ übernimmt, abhängig davon, in welcher Funktion sie aufgerufen wird, unterschiedliche Aufgaben.
Erfolgt der Aufruf in der Funktion „startbutton.m“ (Übergabeparameter: 0, 0, 1, GUI-Identifier) wird zu Beginn die Funktion „mapinput.m“ aufgerufen, die für das Einlesen des Pfad zum Map-File sorgt. In weiterer Folge wird die angegebene Datei auf Existenz kontrolliert und die Variablen bzw. ihre Adressen entsprechend der Syntax (siehe Abschnitt 2.3.1) in einen „Container“ geladen. Aus diesem Grund ist auch zwingend eine MATLAB-Version ab R2009a notwendig, da Container erst ab dieser Version unterstützt werden! Die im Container eingetragenen Variablen können im GUI aufgelistet und für die Datentransfer-Verbindung vorbereitet werden.
Beim Start einer Datentransfer-Verbindung werden zu Beginn die Indizes und die Adressen der Variablen sowie Softauge-Kanäle übermittelt. Zwischen der Auswahl des Map-Files beim Projektstart und dem Start der Verbindung kann sich jedoch die Firmware und somit das zugehörige Map-File ändern. Um sicher zu stellen, dass eine korrekte Adresse für die im GUI aufgelisteten Variablen verwendet wird, erfolgt zu Beginn der Funktion „RXon.m“ ein Aufruf der Funktion „map2container.m“ (Parameter: 0, 0, 0, GUI-Identifier). Der Aufruf dient jedoch in diesem Fall nicht dazu, ein neues Map-File einzulesen, sondern mit Hilfe des bereits hinterlegten Map-File-Pfad alle Adressen und Variablen erneut in den Container zu laden. Dadurch wird sichergestellt, dass die Adressen der Variablen stets aktuell sind. Weiters wird mit Hilfe der Funktion „MAPcheck.m“ kontrolliert, ob die im GUI aufgelisteten Variablen noch im aktuellen Map-File vorhanden sind.

3.4.2 Funktion mapinput.m

Die Funktion „mapinput.m“ dient zum Einlesen des Pfad zum Map-File und wird gegebenenfalls durch die Funktion „map2container.m“ aufgerufen.
Diese Funktion stellt sicher, dass die ausgewählte Datei eine *.map-Datei ist und gibt bei Bedarf eine Fehlermeldung aus.
Als Übergabeparameter für diese Funktion wird das Handle der GUI-Figure benötigt, der Rückgabewert dieser Funktion entspricht dem eingelesenen Pfad.

3.4.3 Funktion MAPcheck.m

Mit Hilfe der Funktion „MAPcheck.m“ wird kontrolliert, ob die im GUI aufgelisteten Variablen bzw. die einem Softauge-Kanal zugeordneten Variablen noch im Map-File vorhanden sind. Zudem sorgt diese Funktion für eine erneute Zuweisung der Map-File-Adressen an die in der Struktur gesicherten Variablen.
Eine Ausnahme bilden dabei Strukturvariablen im GUI. Diese können einzig anhand der Existenz ihrer „Structurparents“ überprüft werden, da nur diese im Map-File aufgelistet sind (siehe Abschnitt 2.3.5.2). Auch die Aktualisierung der Adressen von Strukturelementen erfolgt durch eine Neuberechnung auf Basis der im Map-File aufgelisteten Parentvariablen und nicht durch eine einfache Zuweisung.
Falls eine im GUI aufgelistete Variable bzw. die Parentvariable eines Strukturelemts nicht mehr im aktuellen Map-File vorhanden ist, wird die entsprechende Variable markiert und durch die Funktion „makelist.m“ im GUI rot eingefärbt. Zudem erscheint eine darauf hinweisende Fehlermeldung.
Aufgerufen wird diese Funktion gegebenenfalls in der Funktion „map2contianer.m“ mit einem Handle der GUI-Figure als Übergabeparameter.

3.5 Funktionen für die Kommunikation

3.5.1 Allgemeines

Bei allen Funktionen, die eine Kommunikation mit dem DSP durchführen, wird die Verbindung zum DSP mit Hilfe einer „Try-Catch-Funktion“ überwacht. Dadurch werden Fehler abgefangen, die durch ein Verbindungsproblem ausgelöst werden.
Der Empfang eines Telegramms und die neuerliche Anforderung einer DSP-Antwort bei nicht bestandener Prüfsummenprüfung wird bei allen betreffenden Funktionen durch eine while-Schleife realisiert. Bei korrektem Empfang der DSP-Antwort wird diese abgebrochen, andernfalls wird eine Antwort durch die while-Scheife noch einmal angefordert.
Sollte die Antwort, trotz mehrmaliger Anforderung, nicht innerhalb eines vorgegebenen Zeitraumes (siehe Tabelle 2.5, Variable „Checksum_timeout“) fehlerfrei eingetroffen sein, wird die while-Schleife ebenfalls unterbrochen und eine Fehlermeldung ausgegeben.
Die Messung des Zeitraumes vom Start der while-Schleife bis zum Empfang einer DSP-Antwort wird mit Hilfe der Befehle „tic“ und „toc“ realisiert.

3.5.2 Funktion DSPinit.m und connection_set.m

Mittles der Funktion „connection_set.m“ wird ein Verbindungs-GUI erstellt, mit dem man die Art und die Parameter einer Verbindung auswählen kann. Aufgerufen wird diese Funktion über den Verbindungs-Button in der Menüleiste des GUI sowie durch die Funktion „savevar.m“. Übergabeparameter sind zwei Dummyvariablen sowie die Nummer der GUI-Figure.
Betätigt man den Button „Initialize DSP“ in diesem Verbindungs-GUI, wird die Callbackfunktion „DSPinit.m“ ausgeführt.
Diese Funktion erstellt und öffnet in MATLAB die gewählte Verbindungsart passend zu den gewünschten Parametern und überprüft, ob der DSP auf ein Initialisierungsprotokoll richtig antwortet.
Um eine IP-Adresse auf deren Existenz überprüfen zu können, können ausgehend von dieser Funktion „Pings“ gesendet werden (siehe Abschnitt 2.3.4).
Ist das Öffnen der Verbindung entsprechend der gewählten Parameter möglich und antwortet der DSP korrekt, kann in weiterer Folge die Datentransfer-Verbindung mit Hilfe der Funktion „RXon.m“ gestartet werden.

3.5.3 Funktion RXon.m

Die Funktion „RXon.m“ bildet die Basis der Datentransfer-Verbindung und wird als Callback des „Connect-Buttons“ im GUI ausgeführt.
Als Übergabeparameter dienen zwei Dummyvariablen und die Figurenummer des GUI, mit deren Hilfe das GUI-Handle geladen wird.

Diese Funktion übernimmt zu Beginn mehrere Aufgaben:

- Das Map-File wird erneut anhand der Funktion „map2container.m“ geladen.
- Die Grafik von „The Eye“ wird mit Hilfe der Funktion „eye_reset.m“ gelöscht.
- Die Adressen der Variablen werden im DSP durch die Funktion „RXTXaddress.m“ einem Index zugewiesen.
- Die Adressen der gewählten Softauge-Variablen und die entsprechenden Zoomwerte werden mittels der Funktion „TXsoftauge.m“ übertragen

Sind alle vier beschriebenen Vorgänge erfolgreich abgeschlossen, startet die eigentliche Datentransfer-Verbindung. Dafür kommt eine while-Schleife zum Einsatz, die bei jedem Durchlauf die benötigte Anzahl an Telegrammen zur Anforderung von Variablenwerten erstellt und die DSP-Antwort auswertet. Die in der Antwort enthaltenen Werte werden im Handle des GUI gesichert und in den entsprechenden Feldern der Variablenzeilen eingetragen. Außerdem wird nach einem erfolgreichen Auslesevorgang bei Bedarf die Grafik von „The Eye“ mittels „eye_plot.m“ aktualisiert.
Ist eine Antwort vollständig verarbeitet, stoppt die while-Schleife vor einem neuerlichen Durchlauf an der Codezeile „pause(delay);“ für den am Schieberegler vorgegebenen Zeitraum. Erst danach wiederholt sich der oben beschriebene Vorgang.
Soll ein Callback, beispielsweise für das Senden eines Variablenwertes, ausgeführt werden, so stoppt die while-Schleife erst an der im Ablauf nächstgelegenen Pause-

Stelle. Ist das Callback abgearbeitet, nimmt die while-Schleife ihren Durchlauf wieder an der Pause-Stelle auf.
Tritt ein Fehler während der Kommunikation auf, auch innerhalb eines Callbacks, wird die bestehende Verbindung gelöscht und durch „RXon.m“ eine Fehlermeldung ausgegeben.

3.5.4 Funktion RXTXaddress.m

Wie bereits in Abschnitt 2.2.5 erläutert, werden die Werte der Variablen nicht direkt über deren Adresse ausgelesen bzw. manipuliert, sondern indirekt über einen Index. Dieser Index beschreibt eine Registerposition im Adressenregister des DSP, in dem die Adresse zuvor abgelegt worden ist.
Im GUI können maximal 24 Variablen aufgelistet werden. Aus diesem Grund ist die Anzahl der im DSP gleichzeitig benötigten Registereinträge auf 24 beschränkt. Um die Verwaltung der Indizes einfach zu gestalten, bietet sich an, dass die Adresse der in der ersten Zeile dargestellten Variable dem Index 1 zugewiesen wird, die zweite Zeile dem Index 2 usw. Somit repräsentieren die Positionen der einzelnen Variablenzeilen, beginnend bei der obersten Zeile, den Index der Variable im DSP.
Wird nun eine Datentransfer-Verbindung aufgebaut, übermittelt die Funktion „RXon.m“ mit der Funktion „RXTXaddress.m“ der Reihe nach alle Adressen der aufgelisteten Variablen.
Kommt es während einer bestehenden Datentransfer-Verbindung zu einer Änderung der Auflistung, z.B. durch hinzufügen, löschen oder ändern einer Variable, wird ebenfalls die Funktion „RXTXaddress.m“ ausgeführt. Hierbei ist es jedoch ausreichend, dass nur die jeweils betroffenen Indizes aktualisiert werden.
Um der Funktion „RXTXaddress.m“ mitzuteilen, welche Indizes aktualisiert werden müssen, wird als erster Übergabeparameter eine Matrix mit den zu aktualisierenden Zeilennummern definiert. Anhand des zweiten Übergabeparameter erhält die Funktion das Handle der GUI-Figure.
Da sich das GUI-Handle während dem Durchlauf der Funktion ändert, wird dieses als Rückgabewert an die aufrufende Funktion zurück gegeben.

3.5.5 Funktion TXsoftauge.m

Die Funktion „TXsoftauge.m“ hat die Aufgabe, dem DSP die Kanäle mit den Adressen der gewählten Softauge-Variablen sowie die zugehörigen Zoomwerte zu übermitteln.
Ausgeführt wird diese Funktion zu Beginn von „RXon.m“ für alle vier Kanäle. Sollte während einer bestehenden Datentransfer-Verbindung die Variable eines Kanals oder ein Zoomwert geändert werden, wird ebenfalls diese Funktion ausgeführt.
Als Übergabeparameter werden das Handle der GUI-Figure sowie die zu aktualisierenden Kanäle in Form einer Matrix von der aufrufenden Funktion übergeben.
Als Rückgabewert wird der aufrufenden Funktion das aktualisierte Handle zurück gegeben.

3.5.6 Funktion TX.m

Die Funktion „TX.m“ wird als Callback des Buttons „Send“ oder bei Bestätigung eines in das GUI eingetragenen Wertes mit „Enter“ ausgeführt, sofern die Datentransfer-Verbindung aktiv ist.
„TX.m“ dient zur Manipulation von Variablenwerten im DSP und ist in mehrere Bereiche aufgeteilt. Zu Beginn wird das entsprechende Protokoll erzeugt, das im Anschluss gesendet wird. Bei erfolgreicher Übertragung werden zum Abschluss die Färbungen der „write-Felder“ aktualisiert (siehe Abschnitt 2.3.5.4).
Zur Übertragung wird der im GUI dargestellte Hexadezimalwert in dezimaler Form herangezogen, da dieser bereits in Zweierkomplementdarstellung vorliegt.

3.5.7 Funktion ramread.m und ramload.m

Wird der RAM-Button in der Menüleiste des GUI betätigt, wird als Callback die Funktion „ramread.m“ ausgeführt und eine eventuell bestehende Datentransfer-Verbindung unterbrochen.
Diese Funktion fordert zur Eingabe eines Pfad und eines Namen für die mat-Datei auf, in der die RAM-Variablen gespeichert werden sollen.
In weiterer Folge erstellt die Funktion ein RAM-GUI, indem die gewünschte Blockgröße des RAM’s und die gewünschte Anzahl an auszulesenden Blöcken festgelegt werden kann.
Durch die Betätigung des Buttons „LOAD RAM“ wird die Callbackfunktion „ramload.m“ ausgeführt, die zwei Aufgaben ausführt.

- Zuerst wird an den DSP eine Initialisierung gesendet, um die Variablen des RAM in die gewünschte Blockgröße zusammenzufassen.
- Konnte die Initialisierung erfolgreich durchgeführt werden, wird eine for-Schleife für die Anzahl an gewünschten Variablenblöcken durchlaufen. In jedem Durchlauf wird der Block mit jenem Index aus dem RAM angefordert, der dem aktuellen Index der for-Schleife entspricht.

Konnte der gesamte Vorgang fehlerfrei durchgeführt werden, werden die empfangenen Variablenblöcke in der zuvor festgelegten Datei gespeichert.
Tritt ein Verbindungsfehler auf, erstellt die Funktion „ramload.m“ ein Fenster mit drei Optionen zur Fehlerverarbeitung. Zur Auswahl stehen:

- Ladevorgang komplett abbrechen.
- Ladevorgang abbrechen und RAM-GUI anzeigen um Blockgröße und Blockanzahl zu ändern.
- Ladvorgang nach dem zuletzt korrekt empfangenen Block wieder fortsetzen. Dafür wird der Index der for-Schleife mit einem entsprechenden Offset versehen und wieder gestartet.

Eine eventuell zuvor bestehende Datentransfer-Verbindung könnte am Ende der Funktion „ramload.m“ wieder gestartet werden. Dies würde jedoch die Performance des gesamten Programms beeinflussen, da aufgrund der while-Schleife in der Funktion „RXon.m“ eine Verschachtelung zwischen dieser Funktion und „ramload.m“ stattfinden würde.

3.6 Variablen-Auswahlfenster

3.6.1 Funktion variable_list.m

Die Funktion „variable_list.m“ ist verantwortlich für die Erstellung des GUI zur Variablenverwaltung.
Diese Funktion dient als Callback für den Variablen-Button in der Menüleiste des MATLAB-GUI und wird gegebenenfalls auch innerhalb der Funktion „startbutton.m“ ausgeführt.
Als Übergabeparameter sind die ersten beiden Variablen wieder Dummyvariablen, der dritte Parameter entspricht der Nummer der Figure des MATLAB-GUI und dient zum Laden des GUI-Handle.
Die Aufgabe dieser Funktion bezieht sich prinzipiell auf die Erstellung des Fensters zur Variablenverwaltung, in dem zwei Tabellen vorhanden sind. Die linke Tabelle zeigt alle im Map-File vorhandenen Variablen, die rechte Tabelle beinhaltet alle im MATLAB-GUI aufgelisteten Variablen des Map-Files inklusive Informationen über Datentyp und Anzahl an Strukturvariablen.
Wird in die linke Tabelle ein String eingegeben, wird dieser oberhalb der Tabelle dargestellt und die Funkion „fokusliste.m“ ausgeführt. Anhand dieser Funktion werden nur mehr jene Variablen in der Tabelle dargestellt, deren Namen mit dem String beginnen.

3.6.2 Funktion savevar.m

Die Variablenverwaltung wird mit dem Button „OK“ abgeschlossen. Dadurch kommt es zur Ausführung der Callbackfunktion „savevar.m“.
Anhand dieser Funktion wird die Variablenstruktur im Handle der MATLAB-GUI-Figure aktualisiert und das Fenster zur Variablenverwaltung geschlossen.

3.7 „The Eye“

3.7.1 Funktion the_eye_figure.m

Die Funktion „the_eye_figure.m“ erstellt die Figure von „The Eye“ mit allen grafischen Objekten und legt die Stile für die einzelnen Kurven in der Grafik fest.
Diese Funktion wird bereits beim Start des MATLAB-GUI innerhalb der Funktion „IEAM_Com.m“ ausgeführt. Als Übergabeparameter dient das Handle der MATLAB-GUI-Figure.
Da „the_eye_figure.m“ das übertragene Handle mit weiteren spezifischen Elementen ergänzt, dient als Rückgabewert an die aufrufende Funktion das aktualisierte Handle.
Für „The Eye“ wurde festgelegt, dass bis zu sieben Variablen gleichzeitig dargestellt werden können. Der Grund für diese Beschränkung liegt darin, dass diese Anzahl mit verschiedenen Farben gut unterscheidbar ist. Zudem wirkt sich eine größere Anzahl sehr stark auf die Aktualisierungsrate des MATLAB-GUI aus.
Innerhalb dieser Grundfunktion von „The Eye“ wird auch eine 7x8-Zelle erstellt, die als Vorbereitung für die ausgewählten Variablen dienen soll.

Jede Zeile stellt eine Variable dar, jede Spalte repräsentiert eine Eigenschaft:

- Spalte 1: Name der Variable
- Spalte 2: empfangene Variablenwerte
- Spalte 3: Faktor
- Spalte 4: Verwendung eines Schwellwert? (true/false)
- Spalte 5: Schwellwert
- Spalte 6: Zeitpunkte, zu denen die Variablenwerte empfangen wurden
- Spalte 7: Schwellwert bereits überschritten? (true/false)
- Spalte 8: Log-Datei erstellen? (true/false)

Wird eine Variable anhand der Checkbox in der Tabelle von „The Eye“ ausgewählt, wird diese in die Zelle eingefügt.

3.7.2 Funktion eye_plot.m

Durch die Funktion „eye_plot.m“ werden die Daten der ausgewählten Variablen in „The Eye“ eingefügt. Aufgerufen wird diese Funktion in „RXon.m“ nach dem Empfang und der Auswertung eines Telegramms. Als Übergabe- und Rückgabeparameter dient jeweils das Handle des MATLAB-GUI.

Die Funktion teilt sich in folgende Aufgabenbereiche auf:

- Zu Beginn wird die Zeitdifferenz berechnet, die seit dem Erstellen des Referenzzeitpunkts vergangen ist. Dieser Referenzzeitpunkt wird beim Start von „The Eye“ erstellt.
- Im Anschluss wird mit Hilfe einer for-Schleife jede ausgewählte Eye-Variable in der Variablenauflistung des MATLAB-GUI gesucht und der zu diesem Zeitpunkt aktuelle Variablenwert aus dem GUI bzw. aus Variablenstruktur des GUI-Handle übernommen.
- Danach wird kontrolliert, ob die berechnete Zeitdifferenz bereits größer als die maximal darzustellende Zeitperiode ist.
- Ist die Zeitperiode überschritten, wird bei Bedarf ein Logfile erstellt und Spalte 2 bzw. Spalte 6 der Zelle für die Eye-Variablen rückgesetzt ([ ]).
- Ist die Zeitperiode noch nicht überschritten, wird der aus dem GUI-Handle übernommene Variablenwert entsprechend der vorgegebenen Parameter (Faktor und Schwellwert) in Spalte 2 und die berechnete Zeitdifferenz in Spalte 6 der Zelle hinzugefügt.
- Der Inhalt von Spalte 6 und Spalte 2 wird letztlich in die Grafik von „The Eye“ eingetragen.

Für die Darstellung der empfangenen Variablenwerte einer Variable würden sich zwei Darstellungsvarianten ergeben:

- Jeder Wert einer Variable gilt als eigener „Plot“, der unter Verwendung der MATLAB-Funktion „hold“ in die Grafik hinzugefügt wird.
- Alle empfangenen Werte und die entsprechenden Zeitpunkte werden jeweils in einer 1xn-Matrix abgelegt. Jede der sieben möglichen Variablen kann mit Hilfe dieser Matrizen durch jeweils einen „Plot“ dargestellt

werden. Da ohne Verwendung des Befehls „hold“ neue Plots die Alten ersetzen, werden stets die aktuellsten Kurven dargestellt.

Da es aufgrund der relativ geringen Abtastrate zu einem Aliasing kommen kann (Nyquist-Shannon-Abtasttheorem), werden die in der Grafik eingetragenen Werte explizit hervorgehoben und nicht mit einer durchgezogenen Kurve verbunden. Diese Art der Darstellung würde durch die erste Variante automatisch erfüllt werden. Aufgrund des hohen Zeitbedarfs dieser Variante kommt jedoch in der Funktion „eye_plot.m“ die zweite Darstellungsvariante zum Einsatz, bei der der Plotstil entsprechend angepasst wird.

3.7.3 Funktion eye_varlist_actualize.m

Anhand der Funktion „eye_varlist_actualize.m“ werden folgende Aufgaben erfüllt:

- Die aufgelisteten Variablen im MATLAB-GUI werden in die Tabelle von „The Eye“ eingetragen.
- Die in der Tabelle mittels Checkbox ausgewählten Variablen werden in die dafür vorbereitete 7x8-Zelle (siehe Abschnitt 3.7.1) eingetragen.

Aufgerufen wird „eye_varlist_actualize.m“ in der Funktion „makelist.m“ und „eye_start.m“ sowie als Callback der Tabelle in „The Eye“.
Als Übergabeparameter dient der erste Wert als Dummy, der zweite Wert enthält Informationen über die Callback auslösende Tabellenzeile und -spalte und der dritte Parameter stellt die Kennung der Figure des MATLAB-GUI dar.

Die Funktion führt folgende Schritte durch:

- Zelle erstellen, in der die markierten Variablen eingefügt werden sollen
- Liste erstellen, in die alle mittels Checkbox markierten Variablen eingetragen werden
- Zelle für Tabelle erstellen, in der alle Variablen des MATLAB-GUI, ausgenommen Structureparents, eingetragen werden
- Vergleich, welche der bereits zuvor markierten Variablen noch im GUI vorhanden sind und diese gegebenenfalls in der neuen Tabelle wieder markieren bzw. in die zu Beginn erstellte Zelle einfügen
- Zelle in Struktur des GUI-Handle sichern
- Liste mit allen gewählten Eye-Variablen erstellen, die in Popup von „The Eye“ eingefügt wird
- betreffende Variable in der Liste des Popup markieren
- Tabelle aktualisieren

3.8 Struktur

3.8.1 Funktion struktur.m

Die Funktion „struktur.m“ ist das Callback des Struktur-Popup im MATLAB-GUI und dient der Anpassung der Variablenstruktur des GUI-Handle für die Verwendung von Strukturvariablen.

Die Funktion hat folgende Aufgaben zu erfüllen:

- Kontrolle, ob gewünschte Anzahl an Strukturvariablen noch in MATLAB-GUI eingefügt werden kann und gegebenenfalls Meldung ausgeben.
- Verschiebung der Variablen innerhalb der Variablenstruktur nach unten, sodass die benötigte Anzahl an Strukturvariablen an der gewünschten Stelle eingefügt werden kann.
- Berechnung der Adresse der einzelnen Strukturvariablen.

3.8.2 Funktion structname.m

Der Name der Strukturvariablen kann manuell im Textfeld „Edit“ des GUI eingetragen werden. Als Callback dieses Textfeldes wird die Funktion „structname.m“ ausgeführt. Diese Funktion sorgt dafür, dass der Variablenname in der Variablenstruktur des GUI-Handle gesichert wird.
Weiters kontrolliert diese Funktion, ob die Strukturvariable, deren Name geändert worden ist, bereits vorher als Softauge definiert wurde.
Ist dies der Fall, wird auch der Name der Softauge-Variable angepasst. Dies geschieht nach dem folgenden Schema:

- Wird eine Strukturvariable als Softauge-Variable definiert, wird ein Vermerk in der Struktur der Softauge-Variablen bei „structur“ eingetragen. (siehe Funktion „IEAM_Com.m“, Variable GUI_handles.softauge(sf).structur).

  Dieser Vermerk enthält Informationen über den Namen des Structurparent, die Position in der Struktur und die Anzahl an Long-Datentypen bis zu dieser Position.

  Zum Beispiel: struktur.test1 ... Datentyp Long
  struktur.test2 ... Datentyp Integer
  struktur.test3 ... Datentyp Long
  struktur.test4 ... Datentyp Integer

  Wird „test3“ einem Softauge-Kanal zugewiesen, stellt sich der Vermerk durch den String „struktur31“ dar. „struktur“ bezeichnet das Structurparent, „3“ gibt die Position in der Struktur wieder, „1“ repräsentiert die Anzahl an „Longs“ bis zur Position 3.

  Anhand dieses Vermerks kann eine Strukturvariable eindeutig identifiziert werden (der Defaultname jeder Strukturvariable ist identisch).

- Die Funktion „stuctname.m“ erstellt nun nach dem erläuterten Prinzip den Softauge-Namen der Strukturvariable.
- Mit Hilfe dieses Namens werden alle vier Softauge-Kanäle überprüft, ob die Strukturvariable in Verwendung ist.
- Wird an einem der Kanäle die Strukturvariable identifiziert, wird der dargestellte Name der Softauge-Variable angepasst.

3.8.3 Funktion longint.m

Die Funktion „longint.m“ ist das Callback der Datentyp-Checkbox im MATLAB-GUI und hat drei wesentliche Aufgaben:

- Durch diese Funktion wird der zu verwendende Datentyp für die entsprechende Variable festgelegt. Eventuell bereits empfangene bzw. in das GUI eingetragene Daten werden gegebenenfalls passend zum neuen Datentyp umgerechnet.
- Adressen von Strukturvariablen werden berechnet, wenn diese durch eine Änderung des Datentyps betroffen sind.
- Die Funktion kontrolliert, ob durch die Änderung des Datentyps einer Strukturvariable die Adresse einer Softauge-Variable angepasst werden muss. Dafür wird nach dem selben Schema vorgegangen, wie bereits in Abschnitt 3.8.2 erläutert.

4. Zusammenfassung

Für eine feldorientierte Regelung von Asynchronmaschine, Synchronmaschine und permanenterregte Synchronmaschine ist eine Verwendung von Umrichtern notwendig. Diese müssen an die jeweilige Maschine, die durch sie betrieben werden soll, angepasst bzw. parametriert werden.
Im Zuge dieser Diplomarbeit ist eine auf MATLAB basierende, grafische Benutzeroberfläche zur Parametrierung von Umrichtern erstellt worden, die über eine serielle Kabelverbindung oder über ein Netzwerk bzw. über das Internet mit dem Umrichter kommunizieren kann. Durch diese Art der Verbindung ergibt sich die Möglichkeit, die Parametrierung räumlich getrennt und über große Distanzen durchführen zu können.
Ist eine Verbindung zwischen Umrichter und Computer aufgebaut, können anhand dieses MATLAB-GUI alle Parameter eingelesen, übersichtlich aufgelistet, manipuliert und mit der Funktion „The Eye“ grafisch abgebildet werden. Weiters bietet dieses MATLAB-GUI die Möglichkeit, Parameter in Gruppen zu verwalten und auch als Struktur darzustellen.
Der Vorteil in der Verwendung von MATLAB liegt darin, dass die empfangenen Parameterwerte bei Bedarf direkt verarbeitet werden können bzw. das GUI im Funktionsumfang einfach erweitert werden kann.
Die vorliegende Dokumentation beschreibt die an die Benutzeroberfläche gestellten Anforderungen und gibt einen Überblick über die verwendete Hardware. Weiters erläutert sie das für die Kommunikation verwendete Protokoll und listet die einzelnen Telegramme auf.
Der wesentlichste Punkt in dieser Arbeit stellt die ausführliche Beschreibung des MATLAB-GUI dar. Die Beschreibung umfasst u.a. die allgemeinen Voraussetzungen für den Einsatz des MATAB-GUI, wie ein Projekt zur Parametrierung gestartet werden kann und welche Funktionen das GUI bietet. In weiterer Folge wird erläutert, wie ein Parameterwert mit Hilfe von „The Eye“ grafisch dargestellt, wie das MATLAB-GUI in seinen Funktionen mit Hilfe der Software-Parameter angepasst und wie die Software vor Manipulation geschützt werden kann.
Das letzte Kapitel dieser Dokumentation bildet eine Übersicht, aufgeteilt in die wesentlichsten Aufgabenbereiche des MATLAB-GUI, über die wichtigsten Funktionen der Software. Anhand dieser Übersicht soll eine Wartung und Erweiterung der Software erleichtert werden.

5. Literaturverzeichnis

Fischer, Rolf: (Elektrische Maschinen)
Elektrische Maschinen, 9. Überarbeitete und erweiterte Auflage, Carl Hanser Verlag München Wien 1995

Brunner, Conrad U. und Nipkow, Jürg: (Energieeffizienz bei Elektromotoren)
Energieeffizienz bei Elektromotoren; in: Bulletin SEV/AES; Hrsg. Electrosuisse; Nummer 5/2007, S. 15 - 18

Bystron, Klaus: (Leistungselektronik)
Leistungselektronik/Technische Elektronik, Carl Hanser Verlag München Wien 1979

Angermann, Anne et al.: (MATLAB-Simulink-Stateflow)
MATLAB-Simulink-Stateflow, Grundlage, Toolboxen, Beispiele, 6. aktualisierte Auflage, Oldenbourg Verlag München 2009