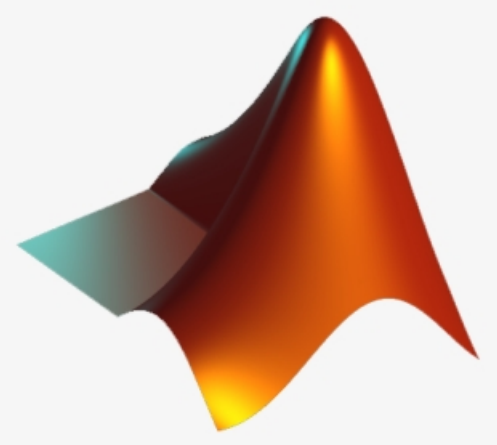
HOCHSCHULE
HAMM-LIPPSTADT

Programmierrichtlinie

für die Erstellung von Software in MATLAB®/Simulink



von:	Prof. Dr. Ulrich Schneider

Stand:	2. Mai 2022
Version:	1.1

Inhaltsverzeichnis

1 Header

1.1 Modul- und Funktionsköpfe

Für die Verständlichkeit der implementierten Software sind Kommentare und Beschreibungen von Modulen und Funktionen unerlässlich. Um einen schnellen Überblick über die vorliegende Implementierung zu erhalten, werden Module durch einen Modulkopf eingeleitet. Dieser befindet sich stets zu Beginn des Moduls. Selbiges gilt für die Beschreibung von Funktionen. Vorlagen für Modul- und Funktionsköpfe werden im Folgenden für verschiedene Programmiersprachen vorgestellt. Die Beispiele können aus diesem Dokument kopiert werden. Wichtig ist jedoch die gewissenhafte Änderung der Kopf-Einträge.

1.1.1 MATLAB®-Code

Auch in MATLAB®-Skripten wird ein Modul mit einem entsprechen Kopf wie in Quelltext 1.1 dargestellt eingeleitet. Dieser muss sich vor weiteren Kommentaren oder Implementierungen befinden, damit er als Hilfetext in MATLAB interpretiert wird. Durch die Eingabe des Befehls "help ModulName" in das *Command Window* der MATLAB-Benutzeroberfläche, soll die Beschreibung des Moduls aufgeführt werden. Dazu wird zunächst der Modulname in Großbuchstaben geschrieben um diesen hervorzuheben. Anschließend erfolgt eine kurze Beschreibung des Zwecks dieses Moduls. Nach einer Leerzeile folgen weitere Modul-Daten, die beim Aufrufen der Hilfefunktion nicht aufgeführt werden sollen. Diese beginnen mit dem Erstellungsdatum des Moduls. Anschließend wird die verwendete MATLAB-Version aufgeführt, um Inkompatibilitäten zwischen Programmen zu vermeiden. Aus diesem Grund werden auch die benötigten MATLAB-Toolboxen aufgelistet. Außerdem werden der Name des Autors, optionale Bemerkungen und das Datum der letzten Änderung dokumentiert.

Quelltext 1.1: Strukturierung eines Modulkopfs in MATLAB

```
%*************************************************************************
%                   Hochschule Hamm-Lippstadt                            *
%*************************************************************************
% Modul           : ModulName.m                                          *
%                                                                        *
% Datum           : 04. Oktober 2013                                     *
%                                                                        *
% Funktion        : kurze Funktionsbeschreibung                          *
%                                                                        *
% Implementierung : MATLAB R2022a                                       *
%                                                                        *
% Req. Toolbox    : Notwendige Toolbox                                   *
%                                                                        *
% Autor           : Max Mustermann                                       *
%                                                                        *
% Bemerkung       : Code-Review noch ausstehend                          *
%                                                                        *
% Letzte Änderung : 02. Mai 2022                                         *
%                                                                        *
%*************************************************************************
```

Bei in MATLAB®implementierten Funktion soll der Kopf ebenfalls Daten als Hilfetext enthalten. Dazu wird der Kopf innerhalb der zugehörigen Funktion unmittelbar unter dem Funktionskopf platziert, wie im Quelltext 1.2 an der Beispielfunktion `addiere` dargestellt. Die Hilfe kann im *Command Window* durch de Befehl `help addiere` aufgerufen werden `help addiere`.

Quelltext 1.2: Beispiel für MATLAB®-Hilfe und Header

```
%   Diese Funktion addiert zwei Zahlen
%
%   Syntax:
%       fErgebnis = addiere(a, b)
%
%   Beschreibung:
%       Es werden die zwei Zahlen 'a' und 'b'
%       beliebigen Datentyps addiert.
%
%   Eingangswerte:
%       a: erster Summand
%       b: zweiter Summand
%
%   Rückgabewerte:
%       fErgebnis: Ergebnis der Addition von 'a' und 'b'
%
%   Beispiel:
%       fErgebnis = addiere(7, 12.583)

%*******************************************************************************
%                    Hochschule Hamm-Lippstadt                                 *
%*******************************************************************************
% Modul            : addiere.m                                                 *
%                                                                              *
% Datum            : 04. Oktober 2013                                          *
%                                                                              *
% Funktion         : Diese Funktion addiert zwei Zahlen                        *
%                    Beispiel für eine MATLAB-Hilfe                            *
%                                                                              *
% Implementierung  : MATLAB R2022a                                             *
%                                                                              *
% Req. Toolbox     : -                                                         *
%                                                                              *
% Autor            : Max Mustermann                                            *
%                                                                              *
% Bemerkung        : Code-Review noch ausstehend                               *
%                                                                              *
% Letzte Änderung  : 02. Mai 2022                                              *
%                                                                              *
%*******************************************************************************
function fErgebnis = addiere(a, b)
  fErgebnis = a + b;
end
```

Der Funktionskopf beginnt mit dem Funktionsnamen und einer kurzen Beschreibung des Zwecks der Funktion. Anschließend wird die Syntax genannt, um die Funktionsdefinition mit allen Parametern zu zeigen. In der Beschreibung wird die Funktion genauer als zuvor beschreiben und Auskunft über die Arbeitsweise beziehungsweise Art der Implementierung gegeben. Des Weiteren werden die Bedeutungen der Übergabeparameter der Funktion erläutert. Ebenso die Rückgabewerte. Abschließend wird ein Beispiel zum Funktionsaufruf gegeben. Der ausgegebene Hilfetext dieses Beispielkopfs ist in Quelltext 1.2 dargestellt. Nach einer Leerzeile folgen wie bereits bei der Beschreibung des Modulkopfs erläutert zusätzliche Daten, die nicht beim Aufruf der Hilfefunktion ausgegeben werden sollen. Nach dem Funktionskopf erfolgt die Implementierung der Funktion.

Quelltext 1.3: Ausgegebener Hilfetext der im Quelltext 1.2 implementierten Funktion.

```
>> help addiere
    Diese Funktion addiert zwei Zahlen

    Syntax:
        fErgebnis = addiere(a, b)

    Beschreibung:
        Es werden die zwei Zahlen 'a' und 'b'
        beliebigen Datentyps addiert.

    Eingangswerte:
        a: erster Summand
        b: zweiter Summand

    Rückgabewerte:
        fErgebnis: Ergebnis der Addition von 'a' und 'b'

    Beispiel:
        fErgebnis = addiere(7, 12.583)
```

1.1.2 Simulinkmodell

In Programmen, die in Simulink umgesetzt werden, sollen ebenfalls Köpfe vorhanden sein. Innerhalb eines Simulink Modells können per Doppelklick in das Modell Textfelder zur Kommentierung eingefügt werden wie in Abbildung 1.1 veranschaulicht. Der Kopf aus Quelltext 1.4 wird in ein Textfeld im oberen linken Bereich des Modells eingefügt.

Quelltext 1.4: Strukturierung eines Modellkopfs in Simulink.

```
Modell          : ModellName.slx

Datum           : 04. Oktober 2013

Funktion        : Zweck dieses Modells

Implementierung : Simulink 8.9 (R2017a)

Req. Toolbox    : Example Toolbox

Autor           : Mustermann, Max

Bemerkung       : Code-Review noch ausstehend

Letzte Änderung : 04. Mai 2018
```

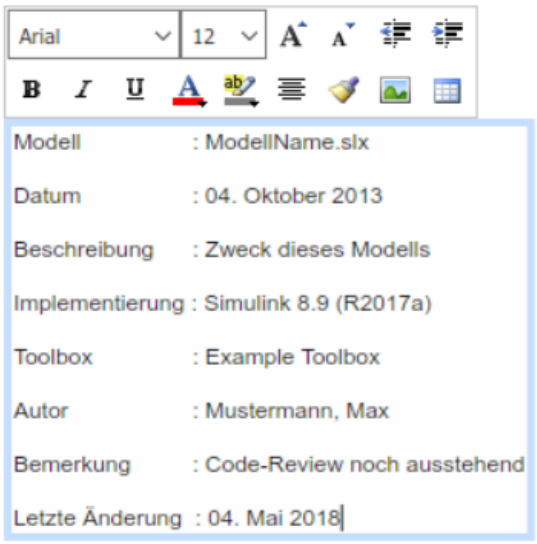


Abbildung 1.1: Kommentarfeld in einem Simulink-Modell mit eingefügtem Kopf aus Quelltext 1.4.

2 Programmierrichtlinien für MATLAB

2.1 Richtlinien für Bezeichner

- Bezeichner sind natursprachliche oder problemnahe Namen oder verständliche Abkürzungen solcher Namen.
- Kein Bezeichner beginnt mit einem Unterstrich
- Generell ist Groß-/Kleinschreibung zu verwenden.
- Zwei Bezeichner dürfen sich nicht nur bezüglich der Groß/Kleinschreibung unterscheiden.
- Die Länge des Bezeichners sollte zwischen 5 und 15 Buchstaben sein.
- Für Laufvariablen in for-Schleifen und als Array-Indizes sind die Kleinbuchstaben `i,j,k,...` zu verwenden.
- Bestehen Bezeichner aus mehreren Worten, dann beginnt jedes Wort mit einem Großbuchstaben, z.B. `AnzahlWorte`.
- Funktionen und Operationen führen Aktionen aus, daher sollte der Name ein bezeichnendes Verb enthalten.
  Beispiel: `Sortiere(anZahlen);`
- Besteht die Wirkung einer Operation nur im Lesen bzw. Schreiben von Instanzvariablen, dann sollte der Name mit `Lese` bzw. `Schreibe` beginnen.
  Beispiel: `Wert=LeseXKoordinate();`
- Variablen sind detailliert zu beschreiben, z. B. `nZeilenZaehler`, `chDateiStatus,...`
- Bei Variablen vom Typ `bool` sollte jeder Bezeichner eindeutig angegeben sein. Beispiel: `bRichtig, bOffen, ...`

2.2 Folgende Präfixe besitzen eine spezielle Bedeutung

gl-	globale Variable
const-	Konstanten
s-	Structs
a-	Array
h-	Handles auf Matlab-Objekte

2.3 Richtlinien für Bezeichner

ganze Zahlen	n-
Geitkommazahlen	f-
boolean Expressions	b-
einzelne Zeichen	ch-
Zeichenketten	st-

MATLAB®unterscheidet nicht zwischen ganzen Zahlen oder Gleitkommawerten, eine Wertzuweisung zwischen ihnen ist problemlos möglich. Aus Gründen besserer Verständlichkeit sollten die oben genannten Präfixe dennoch verwendet werden.

Mehrfacheigenschaften sind entsprechend aufzubauen

an-	Array of int
as-	Array of Struct
ca-	Cell Array

2.4 Richtlinien für die Formatierung

Leerzeichen

- Bei Operatoren werden Operanden und Operator durch jeweils ein Leerzeichen getrennt.
  Beispiel: `nZahl1 + nZahl2 * 3;`

- Zwischen Funktionsname und Klammer steht kein Leerzeichen. Nach der öffnenden und vor schließenden Klammer steht ebenfalls kein Leerzeichen.
  Beispiel: `SetzeBit(nZustand,3);`

Leerzeilen

- Leerzeilen sollen Blöcke von logisch zusammenhängenden Anweisung trennen. Der Deklarationsteil soll stets durch eine Leerzeile von den Anweisungen getrennt werden.
- Jeder Funktionsdeklaration soll eine Leerzeile vorausgehen.
- Umfangreiche Kontrollstrukturen sind durch eine Leerzeile zu trennen.
- Umfangreiche Deklarationen sind durch eine Leerzeile zu trennen.

Einrücken von Strukturen
Alle Strukturen sind entsprechend ihrer `for / if / ... end` einzurücken.

Verwendung von Feldern (Arrays)
Felder sollen immer vollständig initialisiert werden.
Beispiel: `nMatrix=zeros(2,4);`

Variablendeklaration
Variablen sollten zu Beginn eines Blocks initialisiert werden.

Verwendung von Konstanten
Symbolische Konstanten werden durchgehend mit Großbuchstaben geschrieben, wobei längere Bezeichnungen zur besseren Lesbarkeit durch Unterstriche gegliedert werden.
Beispiel: `SAMPLES_PRO_PERIODE`

2.5 Kommentare

Alle Variablen- und Konstantendefinitionen müssen einen kurzen Kommentar zur Beschreibung der Verwendung beinhalten. Alle Funktionen und Algorithmen müssen einen Kommentar zu ihrer Funktion, sowie eine genauere Spezifikation von Ein- und Ausgabeparametern enthalten. Auch komplexere Codeblöcke innerhalb einer Funktion müssen mit einer kurzen Beschreibung versehen werden. Die Funktionsweise des Codes muss im Großen und Ganzen allein anhand der Kommentare nachvollzogen werden können.

2.6 Erstellung von Signalflussplänen (Matlab/Simulink)

Um eine einfache Lesbarkeit von Signalflussplänen zu gewährleisten, werden folgende Richtlinien aufgestellt, sie Abbildung 3.1:

- Eingänge: Hintergrund gelb
- Ausgänge: Hintergrund grün
- Blöcke, die verstellbare Parameter enthalten: Hintergrund hellblau

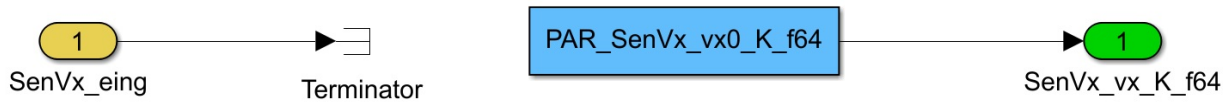


Abbildung 2.1: Einheitliche Darstellung von Ein- und Ausgängen sowie Blöcken mit Parametern.

Alle Parameter (außer physikalische Konstanten, Zahlen wie 1 und 2 etc.) müssen per Parametername angelegt werden. Diese Parameter werden dann außerhalb der Signalflusspläne (Simulink) eingestellt bzw. bedatet.