Programmierrichtlinie für das Erstellen von C-Code

1 Einleitung

Dieses Dokument ist ein Leitfaden für die Namenskonventionen beim Programmieren. Diese Konventionen sind abgeleitet von der ungarischen Notation und entsprechen der Hella Richtlinie HP-GE-232-12.

2 Namenkonventionen von Variablen

Aufbau von Variablen:

Beschreibung	**Modulname**	**Beschreibung**	**_**	**Typ**
Inhalt	Kürzel des Modulnamens. 2 – 5 Buchstaben.	Aussagekräftige Beschreibung des Inhaltes	Unterstrich	u8, s8, u16, s16, u32, s32, st (struct), bit (bit)
Beispiel	**MD_**ErrCount_u8	MD_**ErrCount**_u8	MD_ErrCount**_**u8	MD_ErrCount_**u8**
Bemerkung	projektspezifisch einheitlich, im Datenlexikon definiert. Gilt nicht für lokale Funktionen und lokale Variablen.	Verwendung einheitlicher Bezeichner	Entfällt bei einer Variable die von einem typedef abgeleitet ist.	Entfällt bei einer Variable die von einem typedef abgeleitet ist. Wenn bei der Definition einer struct, enum oder union ein tag deklariert wird, bekommt der Tagname das Suffix tag angehäng t.

2.1 Namen von Funktionen

Funktionsnamen werden durch eine Aneinanderreihung von Wörtern aufgebaut, wobei in jedem dieser Wörter der erste Buchstabe groß geschrieben wird und der Rest klein. Werden

Funktionen außerhalb eines Moduls verwendet, wird das Modulkürzel vorangestellt, handelt es sich um locale Funktionen wird der erste Buchstabe klein geschrieben. Funktionen bekommen keinen Typen-Suffix.

Beispiele:

```
void MD_MisalignmentDetection(void);
static BIT getRawData (void);
```

2.2 Namen von Konstanten

Unter Konstanten werden an dieser Stelle die Definitionen für den Präprozessor (per `#define`) verstanden. Über die Forderung nach einer aussagekräftigen Wahl des Namens hinaus gelten folgende Regeln:

1. Symbolische Konstanten werden durchgehend mit Großbuchstaben beschrieben, wobei längere Bezeichnungen zur besseren Lesbarkeit durch Unterstriche gegliedert werden.

2. Analog zu der Namenskonvention für Variablen wird auch an die Konstante der Typ angehängt. Der angegebene Typ wird dabei durch den Wertebereich vorgegeben, den diese Konstante annehmen kann.

3. Ein Modulpräfix sollte bei globalen Konstanten Aufschluss über den Definitionsort geben.

Beispiel:

```
#define MD_MAX_RAWCHANNEL_TRANSITIONS_u8 ((u8)20) /* globale Konstante */
static const u16 ENOUGH_DATA_LIMIT_u16 = 10;  /* modullokale Konstante */
```

2.3 Namen von Datentypen

Alle vom Benutzer definierten Typen, also diejenigen die mit `typedef` definiert sind, müssen mit der Endung `_t` versehen werden. Der Name von Datentypen wird wie bei Funktionsnamen aus Wörtern gebildet, deren erster Buchstabe groß geschrieben wird.

Beispiel:

```
typedef struct
{
UncertaintyStruct_t Uncertainty_st; // uncertainty of result
s16 Value_s16; // resulting value
}
ResultValueStruct_t;
```

2.4 Namen von Variablen

Diese Notation legt die Benennung der Variablen bezüglich ihres Typs fest. Der Name wird wie bei Funktionsnamen aus Wörtern gebildet, deren erster Buchstabe groß geschrieben wird. Lokale Variablen beginnen mit einem Kleinbuchstaben und globale mit dem Modulpräfix. Als Suffix ist einer der folgenden Typenbezeichner getrennt mit einem Unterstrich zu verwenden:

u8	unsigned Char (8 Bit Integer ohne Vorzeichen)	[0,255]
s8	signed Char (8 Bit Integer mit Vorzeichen)	[−128,+127]
u16	Word (16 Bit Integer ohne VZ)	[0, 65 535]
s16	Word (16 Bit Integer mit VZ)	[−32 768, +32 767]
u32	Double-Word (16 Bit Integer ohne VZ)	[0, 4 294 967 295]
s32	Double-Word (16 Bit Integer mit VZ)	[−2 147 483 648, +2 147 483 647]

Empfehlung (nicht bindend):

bit	Bit (FALSE, TRUE)	[0,1]
p	Zeiger (pointer) auf nachfolgenden Typ	
a	Feld (array) vom nachfolgenden Typ	
st	Struktur	-

Beispiele:

```
u16 MD_StdAlignmentAngle_u16;         /* globale Variable             */
static TrackStruct_t TrackList_ast[25];  /* modulglobales Feld ein Struktur */
static BIT StartupWithDTC1678Active_bit;  /* modulglobale Variable    */
TrackStruct_t* TrackList_pst          /* Zeiger auf eine Struktur     */
BIT channelTransValid_bit;            /* lokale Variable vom Typ BIT  */
```

3 Sonderregeln

3.1 Schleifeninkrement

Array Indices dürfen in einfachen For-Schleifen zur Übersichtlichkeit mit einfachen Zählervariablen belegt werden. In komplexen oder verschachtelten Schleifen sollten jedoch sprechenden Namen verwendet werden. Beispiel:

```
for (i=1;i<3;i++)
{
  Memory[i]=0;
}
```

besser

```
for (index_u8=1; index_u8<3; index_u8++)
{
  Memory[index_u8]=0;
}
```

Weitere Beispiele fur Indexbezeichner sind:

```
cnt_u8, trackCnt_u8, channelNr_u8
```

4 Wortwahl

Die Wortwahl kann durch einige Basisregeln erleichtert werden

- Die Variablen sollten einfach lesbar und verständlich sein. Die Syntax der englischen Worte HorizontalAlignment ist beispielsweise einfacher lesbar als AlignmentHorizontal.
- Lesbarkeit geht vor Kürze . CalculateHorizontalAlignment ist beispielsweise verstandlicher als CalPsi (Ψ ist hier die schwer verständliche Abkürzung für den horizontalen Ausrichtungswinkel).
- Unterstriche dienen nur zur Trennung bei Konstantennamen oder zur Abgrenzung des Datentypen Suffix.
- Bindestriche und andere nicht alphanumerische Zeichen (Buchstaben und Ziffern) sind zu vermeiden.

- Die zuvor beschriebene Variante der ungarischen Notation ist zu verwenden.
- Bezeichnungen, die in Konflikt mit gängigen Bezeichnern von Programmiersprachen stehen, sind zu vermeiden, z. B. `sprintf`, `if`, `for`, `while` etc.

Programmierrichtlinie für das Erstellen von Signalflussplänen (z. B. Matlab/Simulink)

5 Namenskonventionen für Variablen und Parameter

Für die einzelnen Module im Signalflussplan werden Modulkürzel mit meist je 3 Buchstaben verwendet, z. B. für das Modul Lenkwinkelsensor das Kürzel LWS und für das Hauptmodul Sensoren die Kürzel SEN.

Signalnamen enthalten diese Bestandteile, die jeweils durch einen Unterstrich (_) miteinander verbunden werden:

1. Modulkürzel in Groß- und Kleinschreibweise, ggf. mehrere, z. B. SenLws
2. Variablenname, z. B. Lenkwinkel oder auch abgekürzt Lw
3. Koordinatensystem, z. B. _K (körperfest), _V (visionssystem, z. B. Kamera), _I (Inertialsystem), _B (Bahnsystem, tangential an Schwerpunktsbahn)
4. Datentyp wie in 2 beschrieben. In Signalflussplänen meist float mit 64 bit, also _f64 oder für logische Variablen mit den Zuständen 0 oder 1 ein _bit

Beispiel ohne Koordinatensystem (gibt es nicht bei allen Größen): **SenLws_Lw_f64**

Beispiel mit Koordinatensystem: **SenVx_vx_K_f64**

Parameter werden genauso wie Variablen bezeichnet, enthalten aber den Präfix PAR_, so z. B. **PAR_SenLws_Initialwert_u8**

6 Erstellung von Signalflussplänen (Matlab/Simulink)

Um eine einfache Lesbarkeit von Signalflussplänen zu gewährleisten, werden folgende Richtlinien aufgestellt, sie Abbildung 6:

- Eingänge: Hintergrund gelb
- Ausgänge: Hintergrund grün

- Blöcke, die verstellbare Parameter enthalten: Hintergrund hellblau

Bild 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.