**Programmierrichtlinien**

**Robotik**

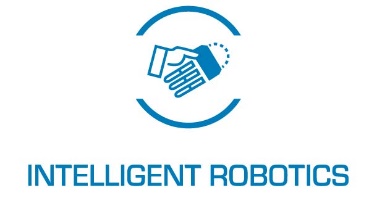
Ersteller: Daniel Kahlig

Version: 2.0

Stand: 01.08.2018

-Vervielfältigung ist nur mit Genehmigung erlaubt.

2018 ESSERT GmbH

****

**Änderungshistorie**

|  |  |  |
| --- | --- | --- |
| **Versionsänderung** | **Datum / Bearbeiter** | **Anmerkung** |
| V1.0 | 20.05.2018 | Entwurf |
| V2.0 | 01.08.2018 | Veröffentlichung |
| V2.01 | 10.04.2019/ Josef Kim | Erweiterung ABB G |
|  |  |  |
|  |  |  |
|  |  |  |

**Ansprechpartner:**

ESSERT GmbH

Großer Sand 18

D-76698 Ubstadt-Weiher

Tel.: +49 7251 32641-00

Fax: +49 7251 32641-99

Email: [info@essert.com](mailto:info@essert.com)

Internet: [www.essert.com](http://www.essert.com)

Inhaltsverzeichnis

[1. Vorwort 5](#_Toc515017936)

[2. Historie – Wozu braucht man einen Software Standard? 5](#_Toc515017937)

[3. Grundlagen der Roboterprogrammierung 5](#_Toc515017938)

[3.1. Das Robotersystem 5](#_Toc515017939)

[3.2. Programmiersoftware 5](#_Toc515017940)

[3.3. Softwarestände 5](#_Toc515017941)

[3.4. Bahnsteuerung 6](#_Toc515017942)

[3.4.1. Punkt-zu-Punkt-Bewegung (PTP) 6](#_Toc515017943)

[3.4.2. Lineare Bewegung (LIN) 6](#_Toc515017944)

[3.4.3. Circular Bewegungen (CIRC) 6](#_Toc515017945)

[3.4.4. Überschleifen von Positionen 6](#_Toc515017946)

[4. Ablauf der Programmerstellung 7](#_Toc515017947)

[5. Programm Recherche 7](#_Toc515017948)

[6. Programm Dokumentation 8](#_Toc515017949)

[7. Richtlinien in der Roboterprogrammierung 8](#_Toc515017950)

[7.1. Allgemeine Variablenbenennung 8](#_Toc515017951)

[7.2. Spezifische Variablenbennung 9](#_Toc515017952)

[7.3. Aufbau der Struktur 9](#_Toc515017953)

[7.4. Aufbau eines Programm 9](#_Toc515017954)

[7.5. Aufbau eines Bewegungsjobs 9](#_Toc515017955)

[7.5.1. …. 10](#_Toc515017956)

[7.5.2. …. 10](#_Toc515017957)

[8. Versionierung der Programme 11](#_Toc515017958)

[9. Schnittstellen Vorlagen 11](#_Toc515017959)

# Vorwort

Diese Richtlinien sind für alle Programmierer Robotik der ESSERT GmbH. Es sollen einheitliche Grundlagen in der Programmierung von unterschiedlichen Robotersystemen zusammengefasst und umgesetzt werden. Da jeder seine eigene Art von Programmierung hat, müssen Standards und einheitliche Vorgaben geschaffen werden. Diese werden in folgenden Seiten definiert.

# Historie – Wozu braucht man einen Software Standard?

In der Vergangenheit wurden die Roboter-Programme durch das Inbetriebnahmepersonal vor Ort erstellt.

Hierdurch entstanden Programme ohne jegliche Struktur, die für andere kaum bzw. nur durch erhöhten Aufwand nachvollziehbar waren (Spaghetti-Code).

Es kam unter anderem vor, dass für gleiche Anlagen unterschiedliche Programme existierten, so daß sich ziemlich hohe Zeiten für die Optimierung bzw. Wartung der Anlagen ergaben.

# Grundlagen der Roboterprogrammierung

## Das Robotersystem

Ein System besteht immer aus einem Roboterarm, einer Robotersteuerung und einem Teachpanel. Die Robotersteuerung bilder das Herzstück eines Roboters. Über das Teachpanel, können wie der Name schon aussagt die Positionen geteacht werden. Außerdem kann ich den Roboter über das Teachpanel manuell verfahren.

## Programmiersoftware

Für jedes Robotersystem gibt es eine Programmiersoftware. Hier habe ich die Möglichkeit die Programme offline vorzubereiten. Außerdem bieten mir die meisten Systeme eine Simulationsoberfläche.

## Softwarestände

Meist befinden sich auf den Steuerungen Softwarestände die uns vom Roboterhersteller vorgegeben werden. Je nach Roboterhersteller sind diese immer enorm wichtig und zu berücksichtigen. Werden Daten mit einem falschen Softwarestand eingespielt kann dies zu erheblichen schäden führen.

## Bahnsteuerung

### Punkt-zu-Punkt-Bewegung (PTP)

PTP steht fur point to point control, also Punkt-zu-Punkt-Steuerung. dazu mussen ¨ nur die anzufahrenden Punkte einer Ruckw ¨ artstransformation unterworfen ¨ werden, zwischen diesen Punkten wird nur mit Achskoordinaten gearbeitet. Bei einer einfachen PTP-Steuerung werden die Sollwerte der Achskoordinaten einfach an die Regelungen ubergeben, so dass die Servomotoren die gew ¨ unschten ¨ Sollwerte anfahren konnen. Dabei ergeben sich folgende Probleme:

* Die Bahn des Greifers zum Zielpunkt ist unkontrolliert und nicht vorhersehbar,
* die auftretenden Geschwindigkeiten und Beschleunigungen hangen von ¨ der Einstellung der Servokreise ab und sind nicht vorhersehbar.

### Lineare Bewegung (LIN)

Fur viele Bewegungen eines Roboters ist eine exakte Einhaltung der gewünschten Bahn notwendig. Das trifft z.B. dann zu, wenn mit der Bewegung eine Bearbeitung des Werkstucks verbunden ist, z.B. Schweißen, Kleben usw. Eine PTP-Steuerung geht hier nicht. Ebenfals ist es notwendig mit linearen Bewegungen zu arbeiten, wenn man ein Bauteil aufnimmt oder ablegt.

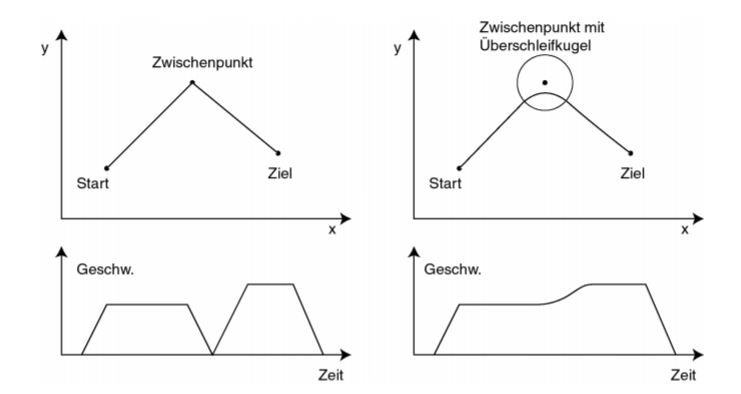
### Circular Bewegungen (CIRC)

Kreisförmige Bewegungen müssen verwendet werden um Konturen abzufahren. Eine Circular Bewegung wird meistens über 3- definierte Teachpositionen festgelegt.

### Überschleifen von Positionen

Freie Positionen wie z.B. Joint- Positionen oder Vor- Positionen werden überschliffen. Die meisten Robotersysteme definieren diesen Abstand mit reach und leave Daten.

Leave= Ab wann darf der Roboter seine Bahn verlassen um den Teachpunkt zu überschleifen

Reach= Ab wann muss der Roboter seine programmierte Bahn wieder erreicht haben

# Ablauf der Programmerstellung

**Folgende Unterlagen werden zur Programmierung benötigt bzw. erstellt:**

* **Ablaufbeschreibung** mit Fehlerstrategie für die Anlage
* **Programmablaufpläne** gemäß Ablaufbeschreibung und Kunden-Anforderung
* **Positionslayout** mit allen Maschinen und benannten Positionen
* **Bewegungsübersicht** des Roboters anhand der Positionen
* **Ein- und Ausgangsbelegung** (Signalname, Beschreibung, Lage des Signals auf der E/A-Karte)
* **"Signal-Schritt-Diagram"** zur Darstellung der Abhängigkeit der Signale untereinander

# Programm Recherche

**Bevor mit der Programmierung begonnen wird, muß überprüft werden:**

* ob bereits ein ähnliches Programm erstellt wurde,
* ob Programmteile von anderen Projekten übernommen werden können,
* oder ob sich in den vorhandenen Datenbanken bereits eine Lösung zu diesem Problem befindet.

**Durch Wiederverwendung von Programmteilen wird die Programmierzeit erheblich verringert.**

**Das Rad muß nicht neu erfunden werden.**

# Programm Dokumentation

Die Inhalte der Programmdokumentation entstehen bereits während der Programmierung.

**Inhalt:**

* Beschreibung der Programmstruktur
* Positionsübersicht / Anlagenlayout
* Ablaufbeschreibung
* Programmablaufplan
* Bewegungsübersicht des Roboters
* Ein- und Ausgangsbelegung
* Übersicht der Datenelemente
* Übersicht und Beschreibung der Vewaltungsroutinen
* Übersicht und Beschreibung der Bewegungsroutinen
* Referenzlisten
* Je nach Programmaufbau besondere Hinweis zur Programmierung, mathematische Grundlagen oder andere Hinweise

# Richtlinien in der Roboterprogrammierung

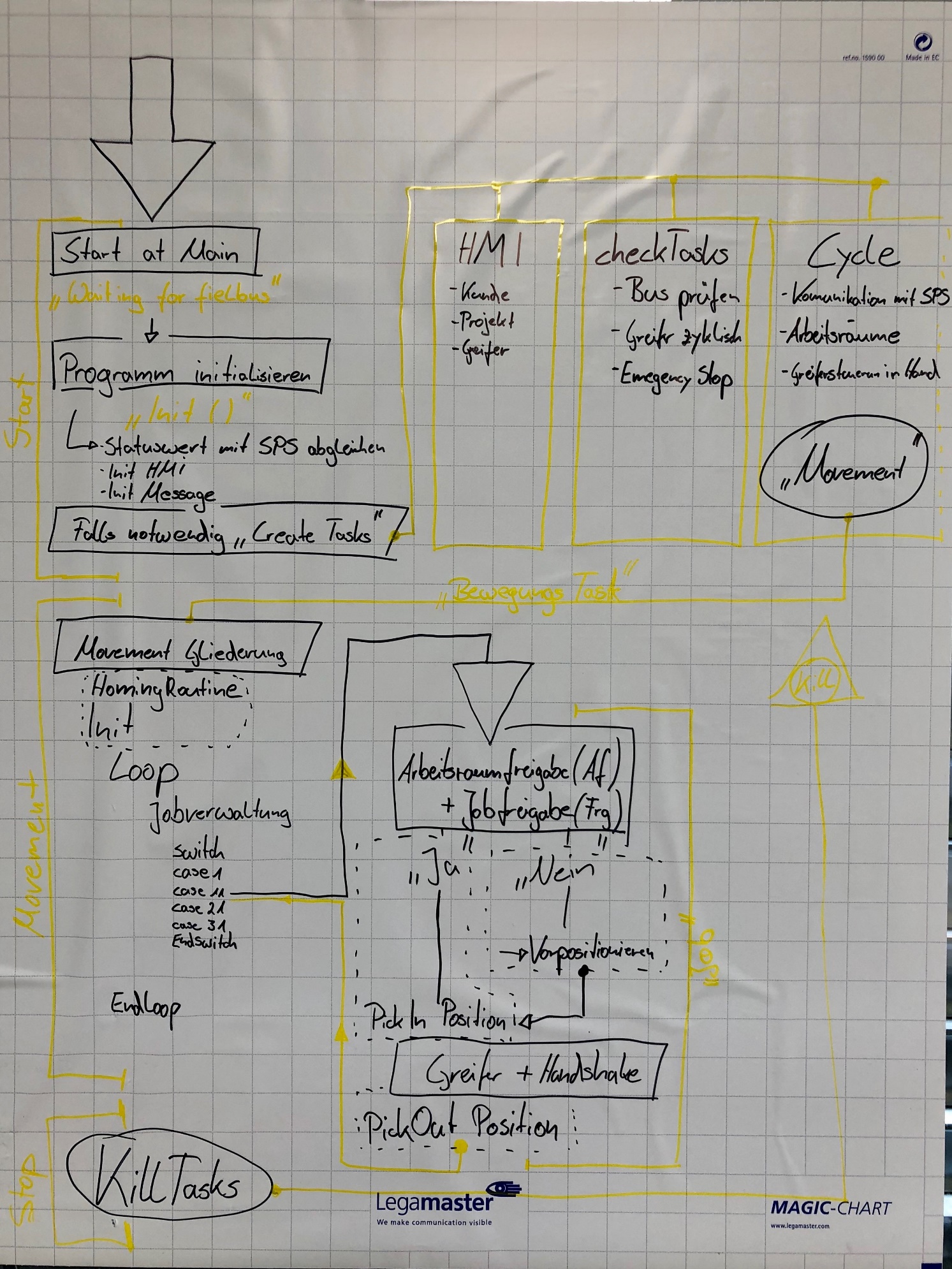
## Allgemeine Variablenbenennung

|  |  |  |  |
| --- | --- | --- | --- |
| **Datentyp Bezeichnung** | **Kürzel** | **Präfix** | **Beispiel** |
| Bool | Bool | b | bCheckOk |
| Uhrzeit/Clock | Clock | ck | ckTimer |
| Numerisch | Num | n | nCounter |
| String | String | st | stBenennung |
| Tooldaten | Tool | t | tSauger |
| Joint Position | Joint | j | jHome |
| Point Position | Point | p | pEndPosition |
| Digitaler Eingang | Input | di | diStartProgramm |
| Digitaler Ausgang | Output | do | doProgrammRunning |
| Eingangsbyte | Input Byte | ib | ibCamOffset |
| Ausgangsbyte | Output Byte | ob | obErrorCode |
| Eingangswort | Input Word | iw | iwCounter |
| Ausgangswort | Output Word | ow | owCounter |

## Spezifische Variablenbennung

|  |  |  |  |
| --- | --- | --- | --- |
| **Datentyp Bezeichnung** | **Kürzel** | **Präfix** | **Beispiel** |
| Koordinatensystem/Werkobjekt/Frame | Frame | f | fPalette |
| Geschwindigkeit/Speeddate/Motiondiscriptor | Motion | m | mFast |
| Überschleifen/Zones/Blending | Zones (reach/leave) | z | zFastSpeed |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Aufbau der Struktur



## Aufbau eines Programm

Unter dem folgenden Link findet man eine Mustervorlage für die Stäubli Programmierung, es werden noch weitere Hersteller folgen:

L:\06\_Intelligent Robotics\Stäubli Robotics\04\_Programm-Vorlage

## Aufbau eines Bewegungsjobs

In diesem Beispiel wird ein Standard Bewegungsjob aufgeführt, es gilt zwischen einen SCARA (4- Achsen) und einen 6- Achsroboter zu unterscheiden.

SCARA (die Logik wird im Roboter Programmiert):

1. (Berechnung der anzufahrenden Position)
2. Abfrage von Arbeitsraumfreigabe und Jobfreigabe
   1. Nicht vorhanden – Anfahren der Vorpositionen und warten bis die Freigaben da sind
   2. Vorhanden – Direkter Fahrbefehl über eine Vorposition zum Endziel
3. (Ggf. muss die Berechnung zur Position nochmals ausgeführt werden)
4. In der Endposition angekommen findet fast immer eine Aktion statt, Greifer / Vakuum usw. nach der Aktion muss geprüft werden ob diese erfolgreich war
5. Nach der Prüfung muss über einen Handshake der SPS mitgeteilt werden das die Aktion erfolgreich durchgeführt wurde
6. Ausfahren auf eine Vorposition
7. Abfragen der Joblogik um in den nächsten Job zu springen oder zurück in die allgemeine Jobverwaltung

6-Achsroboter (die Logik wird von der SPS vorgegeben in der Jobverwaltung):

* Überfahren einer Achsposition (Joint-Position) um den Roboter über die Achstellungen definiert vorzupositionieren
* Abfrage von Arbeitsraumfreigabe und Jobfreigabe
  + Nicht vorhanden – Anfahren der Vorpositionen und warten bis die Freigaben da sind
  + Vorhanden – Direkter Fahrbefehl über eine Vorposition zum Endziel
* (Ggf. muss die Berechnung zur Position nochmals ausgeführt werden)
* In der Endposition angekommen findet fast immer eine Aktion statt, Greifer / Vakuum usw. nach der Aktion muss geprüft werden ob diese erfolgreich war
* Nach der Prüfung muss über einen Handshake der SPS mitgeteilt werden das die Aktion erfolgreich durchgeführt wurde
* Ausfahren auf eine Vorposition
* Überfahren der Achsposition
* Zurück in die Jobverwaltung

Bei einem 6- Achsroboter sollte es möglich sein, alle Jobs dynaschis anzufahren, d.h. die Achspositionen müssen dementsprechend gewählt werden das es immer möglich ist von Achs- zur Achsposition zu fahren. Die Homeposition gehört ebenfalls zu den Achspositionen.

# Versionierung der Programme

Projektnummer\_Kunde\_Projektname\_Versionsnummer

Beispiel:

**4028\_KuttererMauer\_Doppelwandtiegel\_V3.0**

1. Offline Programmierung => V0.x
2. Freigabe von Teamleiter für die Inbetriebnahme => V1.x
3. Nach bestandener Vorabnahme FAT => V2.x
4. Nach bestandener Endabnahme SAT => V3.0
5. Änderungen am Programm nach Endabnahme SAT => V3.x

Diese Versionierungen sind dringend einzuhalten, es wird gewährleistet das jeder Mitarbeiter erkennen kann in welchem Projektstatus sich die Programme befinden.

Die Schnittstellenbeschreibung sollte ebenfalls mit der selben Versionierung versehen werden.

Die Projekte sind regelmäßig im Projektordner abzulegen.

# Schnittstellen Vorlagen

In der Konzeptphase wird in Verbindung zur Steuerungstechnik eine Projektbezogene Schnittstelle ausgearbeitet. Diese dient als Grundlage zur Kommunikation mit der SPS.

Auf unserem Server, sowie in Sharepoint liegen die dafür vorgesehenen Vorlagen zu den jeweiligen Roboterherstellern. Diese sind auf das Projekt anzupassen und im vorhandenen Bit-Muster zu verwenden.

|  |  |  |
| --- | --- | --- |
| **Name** | **Datentyp** | **Beschreibung** |
|  |  |  |
|  |  |  |
|  |  |  |

### ….

|  |  |  |
| --- | --- | --- |
| **Name** | **Datentyp** | **Beschreibung** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### ….

|  |  |  |
| --- | --- | --- |
| **Name** | **Datentyp** | **Beschreibung** |
|  |  |  |