1. Übersicht

Nachfolgendes Dokument enthält die Spezifikation einer Beschreibungssprache für die Konfiguration von Automatisierungssystemen, insbesondere deren I/Os.

Hinweis zu Kompatibilität:
Ausgangspunkt für die hier beschriebenen Sprachen waren die Beschreibungssprachen der Normen IEC 61131-3 und IEC 1499. Für die Definition der System-Konfiguration waren die Konstrukte der Originalsprache nicht ausreichend. Es wurde deshalb eine spezielle Sprache in Anlehnung an die Normen definiert.

Die Sprache arbeitet blockorientiert, d.h. eine Definition besteht immer aus einer Reihe von Beschreibungsblöcken. Jeder Block wird durch ein bestimmtes Schlüsselwort eingeleitet und durch ein weiteres Schlüsselwort wieder beendet. Aus dem einleitenden Schlüsselwort ergibt sich das Format des jeweiligen Blocks. Innerhalb eines Blocks können weitere Blöcke definiert sein. Ein bestimmter Block kann einmalig, optional oder mehrmalig vorkommen.
Bei Bezeichnern wird nicht zwischen Groß- und Kleinbuchstaben unterschieden. Beim Schreiben der Quelldatei werden um die Lesbarkeit zu erhöhen, Schlüsselworte immer komplett mit Großbuchstaben und sonstige Bezeichner komplett mit Kleinbuchstaben geschrieben. Beim Einlesen werden alle Bezeichner in Großbuchstaben gewandelt.

Zum lesen und schreiben der einzelnen Quell-Dateien existiert eine Parser-Bibliothek. Diese Bibliothek kann in die einzelnen Werkzeuge integriert werden und stellt Funktionen zur Verfügung, die es ermöglichen ein Quell-Datei einzulesen und in einer dynamischen Speicherstruktur im RAM abzubilden. Auf dem RAM-Speicherabbild kann der Dateiinhalt dann sehr einfach bearbeitet werden. Ebenso kann mit einer weiteren Funktion aus der Parser-Bibliothek das RAM-Speicherabbild als ASCII-Datei geschrieben werden.

Die Sprache kann auch sehr leicht im XML Format verwendet werden. Die Schlüsselworte werden in diesem Fall einfach als XML-Tags geschrieben.

2. Syntaxnotation

Die Syntax der einzelnen Sprachen wird mit einer Notation beschrieben, die aus einer Menge von terminalen Symbolen, nicht-terminalen Symbolen und Produktionsregeln besteht.
Terminale Symbole sind entweder Schlüsselworte die in Hochkomma (' ') eingeschlossen sind oder Identifer, die die Namenskonventionen erfüllen.
Non-terminale Symbole werden mit Hilfe der Produktionsregeln auf terminale Symbole reduzierter.
Eine Produktionsregeln besteht aus:

non-terminales Symbol ::= erweiterte Struktur

Eine erweiterte Struktur kann aus mehreren Zeilen bestehen die entweder nichts (NIL), terminale Symbole oder nicht-terminale Symbole enthalten. Zusätzlich existieren folgende Notationsformen:

Wenn S eine erweiterte Struktur ist, dann bedeutet:

[S] die erweiterte Struktur kann nicht oder genau einmal vorkommen

{ S } die erweiterte Struktur kann nicht oder n-mal vorkommen

Wenn S1 und S2 erweiterte Strukturen sind, dann bedeutet

S1 | S2 entweder S1 oder S2

S1 S2 Aneinanderreihung von S1 und S2

Mit Klammern kann die Reihenfolge der Auswertung beeinflußt werden.

3. Aufbau und Inhalt der Konfigurations-Beschreibung

3.1 Allgemeines

Alle relevanten Konfigurations-Informationen eines Projekts werden in einer sog. Konfigurations-Datei abgelegt. Zu den Konfigurations-Informationen gehört die Definition von globalen Variablen, die Beschreibung der im Projekt verwendeten Hardwarekomponenten (EAMs, ZEs ...) sowie die Beschreibung der verwendeten Bussysteme und der daran angeschlossenen Geräte.
Die Konfigurations-Datei wird im ASCII-Format gespeichert und hat die nachfolgend beschriebenen Syntax. Die beschriebene Syntax entspricht einer kontextfreien Sprache und kann somit relativ einfach gelesen und geschrieben werden. Das Lesen und Schreiben erfolgt durch einen sog. Sprachparser, der als Library zur Verfügung steht. Der Sprachparser ließt den Inhalt des ASCII-Files ein und legt das Ergebnis in einer definierten Speicherstruktur ab.
Die Sprache besteht aus einer Reihe von sog. Blöcken, die durch ein Schlüsselwort eingeleitet werden. Der Inhalt eines Blocks ist in Abhängigkeit von diesem Schlüsselwort definiert. Ein Block endet immer durch ein Schlüsselwort das folgendermaßen aufgebaut ist: END_keywort
Dadurch ist eine max. Erweiterbarkeit der Sprache gewährleistet. Findet ein Sprachparser einen Block den er nicht kennt, ignoriert er einfach alle Einträge des Blocks bis er das Schlüsselwort für das Blockende findet. Ein minimaler Sprachparser muß die Blöcke SYSTEM und VERSION unterstützen (siehe nachfolgende Syntaxdefinition). Alle weiteren Blöcke sind aus Sicht des Parsers optional und können vorhanden sein oder nicht.
Blöcke können geschachtelt werden, d.h. sie können ihrerseits wieder Blöcke enthalten.

3.2 Syntaxdefinition:

Die Syntax der Quelldatei ähnelt der in den Normen IEC 1131 und IEC 1499 verwendeten Beschreibungssprache. Die Beschreibungssprache unterstützt alle für die Beschreibung eines vernetzten Steuerungssystems erforderlichen Elemente (Device, Bus, Variable, Applikation ...). Die Beschreibung von Details, die häufig auch vom Typ des jeweiligen Elements abhängen (z.B. Art des Geräts, Art des Bussystems), erfolgt mit sog. Attributlisten. Dabei handelt es sich um eine Liste von Zuweisungen die immer aus einem Bezeichner und einem Wert bestehen. Wie der entsprechende Wert zu interpretieren und zu verwenden ist ergibt sich aus dem Namen des Bezeichners und dem Typ des jeweiligen Elements.
Auf diese Weise kann der Beschreibungsumfang der einzelnen Komponenten flexibel erweitert werden, ohne den Sprachumfang dieser Syntaxdefinition verändern zu müssen.

Die Beschreibung eines Systems beginnt mit einem Versionsblock, dessen Attribute Verwaltungsinformationen des Systems enthält. Derzeit ist nur das Attribut SPRACH_VERSION definiert. Es gibt die aktuelle Version der Beschreibungssprache an.
Ein System besteht weiterhin aus zumindest einer Geräte-Instanzbeschreibung (device_configuration) und einer Geräte-Typbeschreibung (device_type_specification). Es müssen min. die Typbeschreibungen zu den instantiierten Geräten vorhanden sein.
Optional kann eine Liste der im System verwendeten Applikationen (application_configuration) vorhanden sein.
Im System vorhandene Bussysteme und die jeweils angeschlossenen Geräte werden ebenfalls in dieser Quelldatei beschrieben (bus_configuration,bus_type_specification).
Die externen Schnittstellen (Busankopplungen, E/As usw.) eines Geräts werden als Interfaces beschrieben. Die verfügbaren Interface-Typen (interface_type_specification) werden ebenfalls hier beschrieben.

```
system_configuration ::=
    'SYSTEM'   system_name ';'
       version_configuration
       device_configuration {device_configuration}
       device_type_specification {device_type_specification}
       [application_configuration]
       {bus_configuration}
       {bus_type_specification}
       {interface_type_specification}
    'END_SYSTEM'

version_configuration ::=
    'VERSION'
       [ parameter_list ]
    'END_VERSION'
```

Die Beschreibung eines einzelnen Geräts erfolgt mit dem Block device_configuration. Er enthält den Namen und den Typ sowie eine Liste der existierenden globalen Variablen, eine Liste der auf diesem Gerät ausführbaren Programme sowie optional eine Liste mit Attributen und eine Liste der auf diesem Device vorhandenen Interfaces.

```
device_configuration ::=
    'DEVICE'
       device_name ':' device_type_name ';'
       {global_variable_list}
       {programm_list}
       [parameter_list]
       {interface_list}
    'END_DEVICE'
```

Jede globale Variable wird mit nachfolgendem Block beschrieben. Die Beschreibung enthält neben dem Variablennamen den Datentyp und eine Liste mit Beschreibungsattributen, die weitere Informationen über die Quelle bzw. das Ziel der Variablen, die Zykluszeit usw. enthalten. Die Bedeutung der Attribute wird über ihre Namen ermittelt. Zusätzlich existiert optional einen Liste mit Blockpositions-Beschreibungen, welche festlegen, in welchem Datenblock diese Variable über ein Bussystem übertragen wird.

```
global_variable_list ::=
    'VAR_GLOBAL' | 'VAR' var_name ':' var_spec_init ';'
       [ parameter_list ]
       { blockposition_list }
    'END_VAR_GLOBAL' | 'EVAR'
```

Die Beschreibung der Blockposition gibt den Namen eines existierenden Datenblocks an. In der Parameterliste wird die Startposition und Länge innerhalb des Blocks beschrieben. Die Blockposition ist eine Möglichkeit, eine globale Variable einem Bustransfer zuzuordnen. Die zweite Möglichkeit besteht darin, bei der Beschreibung des Datenblocks eine Liste von Variablen anzugeben. Beide Varianten werden von der Beschreibungssprache unterstützt. Welche letztendlich verwendet wird, hängt von der Implementierung des Konfigurationswerkzeuges ab.

```
blockposition_list ::=
    'BPOS' block_name ';'
       [ parameter_list ]
    'EBPOS'
```

Jedes Interface wird mit nachfolgendem Block beschrieben. Die Beschreibung enthält neben dem Interfacenamen den Typ des Interfaces und eine Liste mit Beschreibungsattributen, die weitere Informationen über das Interface z.B. den Namen des angeschlossenen Bus enthält.

```
interface_list ::=
```

```
    'INTERFACE' interface_name ':' interface_type_name ';'
       [ parameter_list ]
    'END_INTERFACE'
```

Jedes auf einem Gerät ausführbare Programm wird mit nachfolgendem Block beschrieben.
Dem Programmname folgt in Klammern der Name der Programmdatei (dateiname). Dieser wird als String-Konstante (eingeschlossen in Hochkommas) angegeben. Der
Der Modifizierer ist optional und gibt an, ob ein Programm beim Startup, Shutdown ausgeführt werden soll, oder ob das Programm das nach dem Start zunächst aktiv oder inaktiv ist.
Weitere Informationen zu dem Programm werden in der Attributliste aufgeführt.

```
programm_list ::=
    'PROGRAMM'
       programm_declaration
       [ parameter_list ]
    'END_PROGRAMM'

programm_declaration ::=
    programmname ',' '(' dateiname ')' [execution_modifier] ';'

execution_modifier ::=
    'ACTIVE' | 'STARTUP' | 'SHUTDOWN' | 'INACTIVE'

dateiname ::= character_string


parameter_list ::=
    'PARAMETER_LIST' | 'PARA'
       { attrib_declaration }
    'END_PARAMETER_LIST' | 'EPARA'
```

Für alle im System instantiierbaren Geräte existiert eine Typbeschreibung. Der Inhalt der Typbeschreibung orientiert sich an den real vorhandenen Geräten (ZEs, EAMs, Sensoren) und wird ebenfalls mit Attributen beschrieben. Die Bedeutung der Attribute ergibt sich aus ihrem Namen. Optional können globale Variablen angegeben werden, die für dieses Device typisch sind und die immer vorhanden sind.

```
device_type_specification ::=
    'DEVICE_TYPE' device_type_name ';'
       [ parameter_list ]
      { global_variable_list }
    'END_DEVICE_TYPE'
```

Alle in einem System vorkommenden und ablauffähigen Programme können optional in einer Liste aufgeführt werden.

```
application_configuration ::=
    'APPLICATION'
       { programm_name ';' }
    'END_APPLICATION'
```

Die in einer Konfiguration real vorkommenden Bussysteme werden mit dem Eintrag bus_configuration definiert. Die Buskonfiguration enthält neben dem Busnamen und dem Bustype eine Liste aller Geräte die an diesem Bus (bus_device_list) angeschlossen sind. Weiterhin kann optional eine Parameterliste (bus_parameter_list) und eine Liste der für diesen Bus definierten Datenblöcke (named_data_block) vorhanden sein. Die Parameterliste enthält Attribute, welche die Businstanz näher beschreiben (z.B. Baudrate). Die Bedeutung der Attribute ergibt sich aus ihrem Namen. Die Datenblöcke beschreiben eine zusammenhängende Menge von Daten, die gemeinsam über den Bus ausgetauscht werden

```
bus_configuration ::=
    'BUS'
        bus_name ':' bus_type_name ';'
        bus_device_list
        [parameter_list]
        {named_data_block}
    'END_BUS'

bus_device_list ::=
    'DEVICES'
        {device_name ';'}
    'END_DEVICES'
```

Die Beschreibung eines Datenblocks besteht aus dessen Namen und einer Längenangabe. Es folgen optional eine Liste der Geräte die diesen Block senden, eine Liste der Geräte die den Block empfangen oder anfordern und eine Parameterliste für den Datenblock. Weiterhin besteht die Möglichkeit, die mit diesem Block übertragenen Variablen aufzulisten (transfer_variable_list).

```
named_data_block ::=
    'DATA_BLOCK'  block_name '(' integer ')' ';'
        [block_sender_list]
        [block_receiver_list]
        [parameter_list]
        {transfer_variable_list}
    'END_DATA_BLOCK'

transfer_variable_list ::=
    'TVAR'  variable_name ';'
        [parameter_list]
    'ETVAR'
```

Die Listen mit den Geräten die den Block senden bzw. empfangen enthält derzeit nur die Namen der Geräte. Unter Umständen kann es sinnvoll sein, die Syntax so zu erweitern, daß zu jedem Gerät eigene Attribute (z.B. Zykluszeit) angegeben werden können.

```
block_sender_list ::=
    'BLOCK_SENDER'
        { device_name ';' }
    'END_BLOCK_SENDER'

block_receiver_list ::=
    'BLOCK_RECEIVER'
        { device_name ';' }
    'END_BLOCK_RECEIVER'
```

Für alle im System vorkommenden Arten von Bussystemen existiert eine Typbeschreibung. Der Inhalt der Typbeschreibung orientiert sich an den real vorhandenen Bussystemen (LSB, CAN, ...) und wird mit Attributen beschrieben. Die Bedeutung der Attribute ergibt sich aus ihrem Namen.
Optional können globale Variablen angegeben werden, die für diesen Bustyp typisch sind und die immer vorhanden sind.

```
bus_type_specification ::=
    'BUS_TYPE'    bus_type_name ';'
        [ parameter_list ]
        { global_variable_list }
    'END_BUS_TYPE'
```

Für alle im System vorkommenden Arten von Interfaces existiert eine Typbeschreibung. Der Inhalt der Typbeschreibung orientiert sich an den real vorhandenen Interfaces und wird mit Attributen beschrieben. Optional können einzelnen Funktionalitäten eines Interfaces in sog. Subinterfaces gruppiert werden (z.B. für Interfaces die Eingänge und Ausgänge auf einer Karte realisieren).

```
interface_type_specification ::=
   'INTERFACE_TYPE'  interface_type_name ';'
      { subinterface_list }
      [ parameter_list ]
   'END_INTERFACE_TYPE'


subinterface_list ::=
   'SUB_INTERFACE'   subinterface_name ';'
      [ parameter_list ]
   'END_SUB_INTERFACE'
```

Die Beschreibung eines Attributes besteht aus dem Attributnamen gefolgt von dem Zusweisungszeichen und dem Wert des Attributs. Der Wert wird als String-Konstante (eingeschlossen in Hochkommas) angegeben. Eine Attributangabe wird immer mit einem Strichpunkt abgschlossen.

```
attrib_declaration ::=
   attrib_name   attrib_spec_init ';'

attrib_spec_init ::=
   ':=' character_string
```

Die einzelnen Namen dürfen nur aus den Buchstaben A-Z, den Ziffern 0-9 und dem Zeichen _ bestehen. Als erstes Zeichen darf keine Ziffer stehen. Nationale Umlaute dürfen nicht verwendet werden.

```
system_name ::= identifier
device_type_name ::= identifier
var_name ::= identifier
programm_name ::= identifier
bus_name ::= identifier
bus_type_name ::= identifier
block_name ::= identifier
attrib_name ::= identifier
interface_name ::= identifier
interface_type_name ::= identifier
subinterface_name ::= identifier
```

Für device_name gilt als zusätzlich Namensvereinbarungen, daß der Name immer mit einer oder zwei Ziffern enden muß. Der Integer-Wert dieser beiden Ziffern kann als Device-Nummer interpretiert werden.

```
devivce_name ::= identifier
```

In der Konfigurationsbeschreibung werden wie bei den Programmquellen zusätzliche Informationen innerhalb von Kommentaren abgelegt. Damit die Zusatzinformationen von den gewöhnlichen Kommentaren unterschieden werden können werden sie wie bei den Programmquellen innerhalb des Kommentars durch die Zeichenfolge '@!' eingeschlossen.

```
comment ::=
   '(*' comment_body '*)'
```

```
comment_body ::=
      '@!' additional_information '@!' |
   !!! beliebiger Text !!!
```

Verwaltungsinformationen zu der Konfigurationsbeschreibung werden am Anfang der Datei als 'head_description' gespeichert. Ebenfalls am Anfang der Quellcode-Datei kann optional eine Liste der für diese Programmquelle benötigten Typedefinitions-Dateien vorhanden sein.

```
additional_information ::=
   head_description |
   [ definitions_include_list ]

character_string ::=
   ''' { any_character } '''

head_description ::=
   'HEAD' ':' signed_integer ',' signed_integer ',' identifier ','
   identifier ',' signed_integer ',' signed_integer

definitions_include_list ::=
   'DEFINITIONS'
      path_description { ',' path_description }
   'END_DEFINITIONS'

path_description ::= character_string
```

3.3 Attributlisten der einzelnen Blöcke:

Block VERSION:

SPRACH_VERSION:='00';
Enthält die aktuelle Version der Beschreibungssprache.

Block VAR_GLOBAL:
OPERAND:='DW0.0';
Enthält die Angabe auf welchem physikalischen Operanden die Variable liegt.

ARRAY_DIM:=‘8‘;
ARRAY_DIM:=‘6, 50“;
Enthält die Anzahl der Elemente aus denen die Arrayvariable (Feld) besteht. Zur Zeit ist die Anzahl der Dimensionen auf zwei beschränkt.

COMMENT:=‘Textbeschreibung als String‘

INIT_VALUE:='0';
Initialisierungswert einer Variable

BLOCK:='blockname';
Wird die Variable über einen Bus transferiert, wird hier die Referenz auf den Transferblock eingetragen.

ACCESS:=' RW'; ' Read'; 'Write'; 'Hidden';
gibt an, ob und wie auf die Variable zugegriffen werden darf. Der Default ist ' RW' - sie darf also gelesen und geschrieben werden.

STATUS:=‘0‘,‘1‘,‘2‘,‘3‘

1 **kopierte Variable**, das heißt, das sie auf dem Gerät liegt, aber hier nur in die Konfiguration hineinkopiert wurde, der Konfigurator überliest diese Variablen, und lädt immer nur das Original (bei DPR-Vars sind sie beim zugehörigen HC11 zu finden und die Globalbus-Variablen sind im Gerät „Baugruppe“ gespeichert).
2 es handelt sich um eine **Systemvariable**. Diese kann im Konfigurator nicht gelöscht oder verschoben werden. Lediglich der Kommentar und evtl. der Name läßt sich ändern.
3 Kopie & Systemvariable

Block DEVICE:

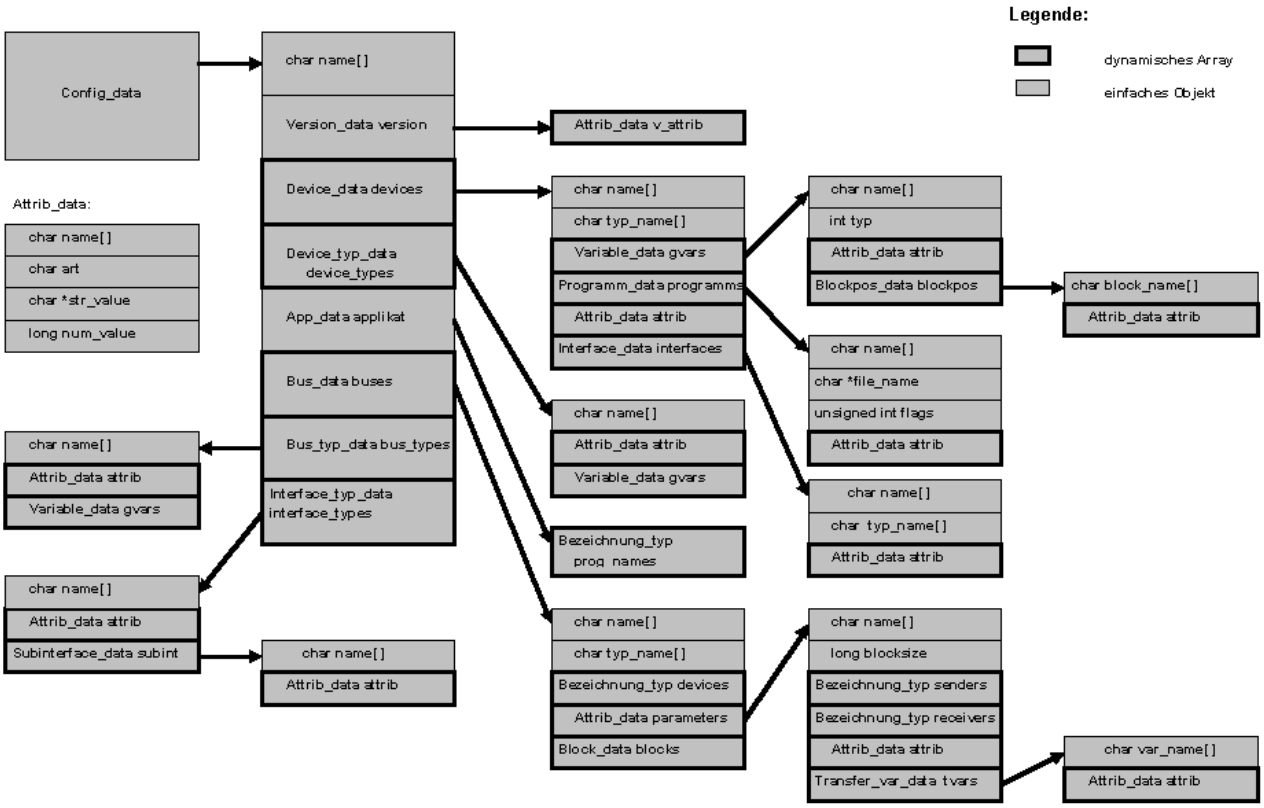
Block DEVICE_TYPE:

Block BUS_PARAMETER:

Block BLOCK_PARAMETER:

Block BUS_TYPE:

3.4 Grafische Darstellung Speicherstruktur Konfiguration:



5. Aufbau und Inhalt der Typdefinitionen

5.1 Allgemeines

Die Definition von bezeichneten Konstanten oder Strukturdatentypen erfolgt in separaten Typedefinitions-Dateien. Die Typdefinition wird im ASCII-Format gespeichert und hat die nachfolgend beschriebenen Syntax. Die beschriebene Syntax entspricht einer kontextfreien Sprache und kann somit relativ einfach gelesen und geschrieben werden. Das Lesen und Schreiben erfolgt durch einen sog. Sprachparser, der als Library zur Verfügung steht. Der Sprachparser ließt den Inhalt des ASCII-Files ein und legt das Ergebnis in einer definierten Speicherstruktur ab.

5.2 Syntaxdefinition:

Die Syntax ist aus den in den Normen IEC 1131 und IEC 1499 verwendeten Beschreibungssprache entnommen. Die Beschreibungssprache unterstützt die Definition von Konstantendefinitionen und Strukturdatentypen.

Eine Typdefinitions-Datei besteht derzeit aus zwei verschiedenen Blocktypen (const_decl_block, data_type_declaration), die jeweils beliebig oft auftreten können.

```
type_definitions_datei ::=
    { const_decl_block }
    { data_type_declarartion }
```

Ein Block mit Konstantendeklarationen enthält Konstantenwerte, die einen eindeutigen Namen haben und beliebig oft verwendet werden können. Durch Änderung des Konstantenwerts in der Definitionsdatei wird er an allen Stellen geändert.

```
const_decl_block ::=
    'VAR' 'CONSTANT'
       const_declaration ';' { const_declaration ';' }
    'END_VAR'

const_declaration ::=
    const_name ':' const_spec_init
```

Ein Block mit Typdefinitionen enthält Struktur-Typen, die einen eindeutigen Namen haben und mehrere Datenelemente unter diesem Namen zusammenbinden. Die einzelnen Elemente müssen als Datentyp einen elementaren Datentyp haben.

```
data_type_declaration ::=
    'TYPE'
       type_declaration ';' { type_declaration ';' }
    'END_TYPE'

type_declaration ::=
    structure_type_declaration

structure_type_declaration ::=
    structure_name ':' structure_declaration

structure_declaration ::=
```

```
    'STRUCT'
       structure_element_declaration ';'
       { structure_element_declaration ';' }
    'END_STRUCT'

structure_element_declaration ::=
    structure_element_name ':' single_element_type_name
```

Die einzelnen Namen dürfen nur aus den Buchstaben A-Z, den Ziffern 0-9 und dem Zeichen _ bestehen. Als erstes Zeichen darf keine Ziffer stehen. Nationale Umlaute dürfen nicht verwendet werden.

```
const_name ::= identifier
structure_name ::= identifier
structure_element_name ::= identifier
```

Auch in einer Type-Definitionsdatei werden zusätzliche Informationen innerhalb von Kommentaren abgelegt. Damit die Zusatzinformationen von den gewöhnlichen Kommentaren unterschieden werden können werden sie wie bei den Programmquellen innerhalb des Kommentars durch die Zeichenfolge '@!' eingeschlossen.

```
comment ::=
    '(*' comment_body '*)'

comment_body ::=
       '@!' additional_information '@!' |
    !!! beliebiger Text !!!
```

Verwaltungsinformationen zu der Type-Definitionsdatei werden am Anfang der Datei als 'head_description' gespeichert.

```
additional_information ::=
    head_description
```