Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Master's Thesis

Think of a cool name for my python XBRL processor

Autumn Term 2023

Supervised by:
Ghislain Fourny
Gustavo Alonso

Author:
Robin Schmidiger

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Your Project Title

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

First name	Last name

Student supervisor(s)

First name	Last name

Supervising lecturer

Roland	Siegwart

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiarism-citationetiquette.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Place and date

Signature

[1]Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Contents

Preface

Bla bla ...

Abstract

Hier kommt der Abstact hin ...

Symbols

Symbols

ϕ, θ, ψ	roll, pitch and yaw angle
b	gyroscope bias
Ω_m	3-axis gyroscope measurement

Indices

x	x axis
y	y axis

Acronyms and Abbreviations

ETH	Eidgenössische Technische Hochschule
EKF	Extended Kalman Filter
IMU	Inertial Measurement Unit
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter

Chapter 1

Introduction

1.1 Motivation

Chapter 2

Einige wichtige Hinweise zum Arbeiten mit LaTeX

Nachfolgend wird die Codierung einiger oft verwendeten Elemente kurz beschrieben. Das Einbinden von Bildern ist in LaTeX nicht ganz unproblematisch und hängt auch stark vom verwendeten Compiler ab. Typisches Format für Bilder in LaTeX ist EPS[1] oder PDF[2].

2.1 Gliederungen

Ein Text kann mit den Befehlen `\chapter{.}`, `\section{.}`, `\subsection{.}` und `\subsubsection{.}` gegliedert werden.

2.2 Referenzen und Verweise

Literaturreferenzen werden mit dem Befehl `\citep{.}` und `\citet{.}` erzeugt. Beispiele: ein Buch [**?**], ein Buch und ein Journal Paper [**?** **?**], ein Konferenz Paper mit Erwähnung des Autors: **?**].
Zur Erzeugung von Fussnoten wird der Befehl `\footnote{.}` verwendet. Auch hier ein Beispiel[3].
Querverweise im Text werden mit `\label{.}` verankert und mit `\cref{.}` erzeugt. Beispiel einer Referenz auf das zweite Kapitel: chapter 2.

2.3 Aufzählungen

Folgendes Beispiel einer Aufzählung ohne Numerierung,

- Punkt 1
- Punkt 2

wurde erzeugt mit:

```
\begin{itemize}
  \item Punkt 1
  \item Punkt 2
\end{itemize}
```

[1] Encapsulated Postscript
[2] Portable Document Format
[3] Bla bla.

Folgendes Beispiel einer Aufzählung mit Numerierung,

1. Punkt 1
2. Punkt 2

wurde erzeugt mit:

```
\begin{enumerate}
  \item Punkt 1
  \item Punkt 2
\end{enumerate}
```

Folgendes Beispiel einer Auflistung,

P1 Punkt 1

P2 Punkt 2

wurde erzeugt mit:

```
\begin{description}
  \item[P1] Punkt 1
  \item[P2] Punkt 2
\end{description}
```

2.4 Erstellen einer Tabelle

Ein Beispiel einer Tabelle:

Table 2.1: Daten der Fahrzyklen ECE, EUDC, NEFZ.

Kennzahl	Einheit	ECE	EUDC	NEFZ
Dauer	s	780	400	1180
Distanz	km	4.052	6.955	11.007
Durchschnittsgeschwindigkeit	km/h	18.7	62.6	33.6
Leerlaufanteil	%	36	10	27

Die Tabelle wurde erzeugt mit:

```
\begin{table}[h]
\begin{center}
 \caption{Daten der Fahrzyklen ECE, EUDC, NEFZ.}\vspace{1ex}
 \label{tab:tabnefz}
 \begin{tabular}{ll|ccc}
 \hline
 Kennzahl & Einheit & ECE & EUDC & NEFZ \\ \hline \hline
 Dauer & s & 780 & 400 & 1180 \\
 Distanz & km & 4.052 & 6.955 & 11.007 \\
 Durchschnittsgeschwindigkeit & km/h & 18.7 &  62.6 & 33.6 \\
 Leerlaufanteil & \% & 36 & 10 & 27 \\
 \hline
 \end{tabular}
\end{center}
\end{table}
```

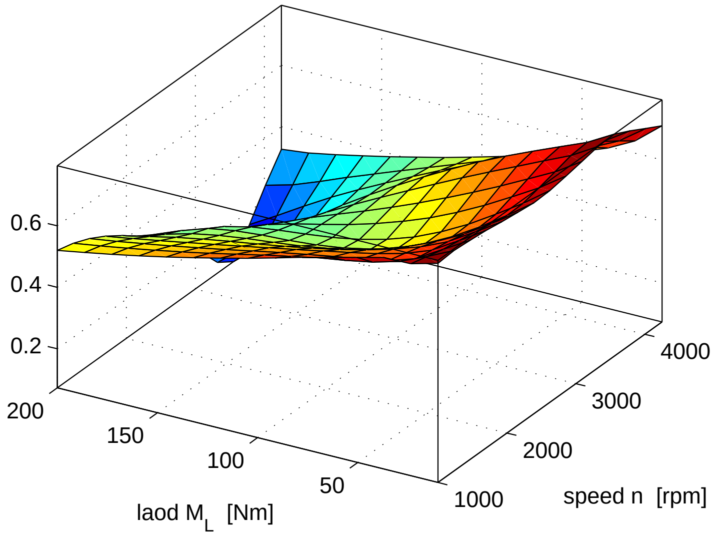


Figure 2.1: Ein Bild

2.5 Einbinden einer Grafik

Das Einbinden von Graphiken kann wie folgt bewerkstelligt werden:

```
\begin{figure}
   \centering
   \includegraphics[width=0.75\textwidth]{images/k_surf.pdf}
   \caption{Ein Bild.}
   \label{fig:k_surf}
\end{figure}
```

oder bei zwei Bildern nebeneinander mit:

```
\begin{figure}
  \begin{minipage}[t]{0.48\textwidth}
    \includegraphics[width = \textwidth]{images/cycle_we.pdf}
  \end{minipage}
  \hfill
  \begin{minipage}[t]{0.48\textwidth}
    \includegraphics[width = \textwidth]{images/cycle_ml.pdf}
  \end{minipage}
  \caption{Zwei Bilder nebeneinander.}
  \label{pics:cycle}
\end{figure}
```

2.6 Mathematische Formeln

Einfache mathematische Formeln werden mit der equation-Umgebung erzeugt:

$$p_{me0f}(T_e, \omega_e) \; = \; k_1(T_e) \cdot (k_2 + k_3 S^2 \omega_e^2) \cdot \Pi_{\max} \cdot \sqrt{\frac{k_4}{B}} \,. \tag{2.1}$$

Der Code dazu lautet:

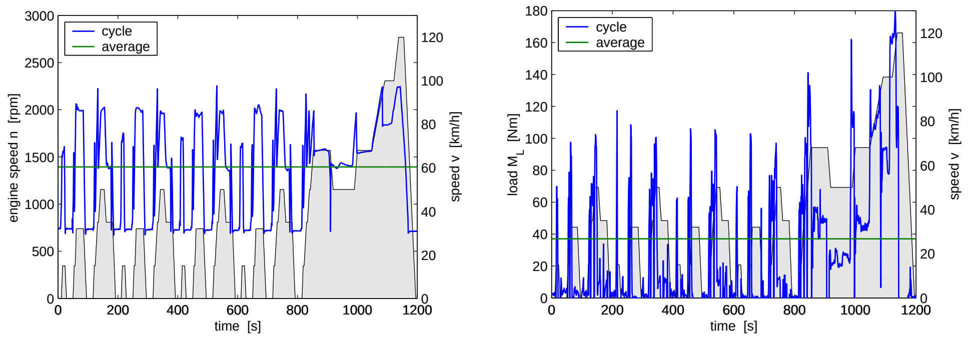


Figure 2.2: Zwei Bilder nebeneinander

```
\begin{equation}
 p_{meOf}(T_e,\omega_e) \ = \ k_1(T_e) \cdot (k_2+k_3 S^2
 \omega_e^2) \cdot \Pi_{max} \cdot \sqrt{\frac{k_4}{B}} \, .
\end{equation}
```

Mathematische Ausdrücke im Text werden mit $formel$ erzeugt (z.B.: $a^2+b^2=c^2$). Vektoren und Matrizen werden mit den Befehlen `\vec{.}` und `\mat{.}` erzeugt (z.B. $\boldsymbol{v}$, $\boldsymbol{M}$).

2.7 Weitere nützliche Befehle

Hervorhebungen im Text sehen so aus: *hervorgehoben*. Erzeugt werden sie mit dem `\epmh{.}` Befehl.
Einheiten werden mit den Befehlen `\unit[1]{m}` (z.B. 1 m) und `\unitfrac[1]{m}{s}` (z.B. 1 m/s) gesetzt.

Chapter 3

Implementation

3.1 Overview

3.2 Facts

A fact is the smallest unit of information in an XBRL report. The word ”Fact” is a term used to describe an individual piiece of financial of business information within an XBRL instance document.
Lets consider a simplified example involving a financial report for a company. Suppose the report contains information about the company’s revenue for the fiscal year ending on December 31, 2022.
In XBRL, a corresponding fact would be represented as follows:

- **Concept:** Revenue
- **Entity:** Microsoft Coorporation
- **Period:** from 2022-04-01 to 2023-03-31 [0]
- **Unit:** USD
- **Value:** 198’000’000 [1]

In this example:

- The **concept** refers *what* is being reported. In this case, ”Revenue” indicates that the fact is reporting information about the company’s revenue.
- The **entity** refers to *who* is reporting. In the case of our example, the entity is ”Microsoft Coorporation”.
- The **period** refers to *when* the information is being reported. The period is defined as the fiscal year 2022.
- The **unit** refers to *how* the information is being reported. In this example, the unit is ”USD”, which indicates that the information is being reported in US dollars.
- The **value** refers to *how much* is being reported. According to Microsoft’s 2022 annual report, the company’s revenue for the fiscal year 2022 was around 198 billion US dollars.

[0] Refers to the fiscal year 2022, which starts on April 1, 2022 and ends on March 31, 2023
[1] https://www.microsoft.com/investor/reports/ar22/index.html

3.2.1 Facts in the XBRL Taxonomy

In the context of XBRL, facts are represented as XML elements in an XBRL instance document.

3.3 Implementation of Facts

3.3.1 Overview

Facts in Brel function very similar to facts in XBRL. However, there are a couple of key differences between XBRL and Brel that are worth highlighting. Before we dive into the details of facts in Brel, let us first look at how facts are represented in XBRL.

3.3.2 Facts in XBRL

In the context of XBRL, facts are represented as XML elements in an XBRL instance document. The following snippet shows an example of a fact in XBRL:

```
<us-gaap:ExtinguishmentOfDebtAmount contextRef="c-216" decimals="-6" id="f-
```

[1]

Thinking back to how we defined facts in chapter TODO, we can see that the above example does not contain all the information that we defined as being necessary for a fact. In particular, the above example does only contain information about the concept, the value, and the unit of the fact. Both the concept and the unit are merely links to other elements in the XBRL taxonomy, but do not contain any information about the concept or the unit themselves. Furthermore, instead of providing information about the entity and the period, the example only contains a reference to a context element.

3.3.3 Contexts in XBRL

In XBRL, contexts are used to provide information about the entity and the period of a fact. If the fact is part of a hypercube, the context also contains information about the dimensions and members of the fact. Contexts are defined in the instance document using the `context` element.
Going back to our example from above, the context element referenced by the fact is defined as follows:

```
<context id="c-216">
    <entity>
        <identifier scheme="http://www.sec.gov/CIK">0000021344</identifier>
        &lt;segment&gt;
            <xbrldi:explicitMember dimension="dei:LegalEntityAxis">ko:Bottl
        &lt;/segment&gt;
    </entity>
    <period>
        <startDate>2023-01-01</startDate>
        <endDate>2023-06-30</endDate>
    </period>
</context>
```

[1] taken from COCA COLA CO's 2023 Q2 report

[2]
The XML snippet fills in some of the missing information from the fact, namely:

- The **entity** refers to *who* is reporting. The entity XML element provides us with a identifier for the entity, as well as information about where the identifier comes from. In this case, the identifier is a Central Index Key (CIK) provided by the US Securities and Exchange Commission (SEC). If we look up the identifier "0000021344" in the SEC's database, we can see that it corresponds to the Coca Cola Company.
- The **period** refers to *when* the information is being reported. As described in chapter TODO, a period can either be a point in time or a duration. In this case, the period is defined as a duration, starting on January 1, 2023 and ending on June 30, 2023.
- The **dimensions** refer to additional information about the fact. This information is only relevant if the fact is part of a hypercube. Dimensions are defined as child elements of the `segment` element, which is a child element of the `entity` element.

A observant reader might have noticed that the placement of dimensions in the context element is not consistent with the placement of both the entity and the period information of a fact. The main reason for this inconsistency is that the XBRL specification was not designed with hypercubes in mind. Hypercubes were added to the XBRL specification at a later point in time.
A second oddity of facts in XBRL is that they are not self-contained. Their definition is spread across multiple elements in the instance document. Most noticably, the context element is defined separately from the fact element. Since different XBRL facts reference the same context element, the incentive behind this design decision is clear: Reducing redundancy. A XBRL instance document can contain thousands of facts, but only a handful of context elements.
However, this design falls apart as soon as we consider the case of hypercubes. To illustrate the problem, consider the following numbers:
In their 2023 Q2 report, the Coca Cola Company

1. reported 1658 facts.
2. defined 13 context elements without dimensions.
3. defined 397 context elements with dimensions.
4. used the same entity for all facts.
5. uesd around 15 different periods for all facts.

So the vast majority of all contexts exist to provide information about the dimensions of a fact. However, the entity and the period information are mostly redundant.

3.3.4 Units in XBRL

In the same way that contexts are used to provide information about the entity and the period of a fact, units are used to provide information about the unit of a fact. Units are also defined in the instance document using the `unit` element. Going back to our example from above, the unit element referenced by the fact is defined as follows:

[2] taken from COCA COLA CO's 2023 Q2 report

```
<unit id=”usd”>
    <measure>iso4217:USD</measure>
</unit>
```

[3]

This XML snippet connects our intuitive understanding of the unit ”usd” with a formal definition of the unit. It does so by referencing the official ISO 4217 currency code for the US dollar.

3.3.5 Concepts in XBRL

As mentioned in chapter TODO, concepts are the fundamental building blocks of XBRL. Each fact is associated with exactly one concept. Unlike units, periods, and contexts, concepts are not defined in the instance document. Instead, they are defined in the XBRL taxonomy, which is a collection of XML schema documents. In case of the above example, the concept is defined in the US GAAP taxonomy, which is a collection of XBRL taxonomies for US Generally Accepted Accounting Principles (GAAP). This already gives us a hint about where to look for the definition of the concept.
The following snippet shows the definition of the concept in the US GAAP taxonomy:

```
<xs:element id=”us-gaap_ExtinguishmentOfDebtAmount” name=”ExtinguishmentOfD
```

[4]

The main purpose of this snippet is to provide a formal definition of the concept and to constrain the values that are allowed for the concept. In particular, it tells us that the concept is a monetary item, that it has a debit balance, and that it has a duration period type.

3.3.6 Facts in Brel

In Brel, facts are represented as Python objects. When processing facts in XBRL, the information about the fact is spread across multiple elements in multiple documents. The main goal of Brel is to provide a more intuitive interface for working with facts. Whereas XBRL facts in XML, concepts, units, periods, etc. are all treated differently, Brel facts treat all aspects of a fact equally.

3.4 Report Elements

In the context of XBRL, report elements are the fundamental building blocks that make up the structure of an XBRL report. Concepts, abstracts, line items, dimensions, members, and hypercubes provide a comprehensive framework for structuring and organizing data in a way that is consistent and machine-readable.

3.5 QNames

Although the motivation behind this XBRL processor is to shield its user from the complexity of XML, we keep one key aspect of XML in our API: QNames.
QNames are a way to uniquely identify an XML element or attribute. They consist of a local name a namespace, which in turn consists of a namespace prefix

[3] taken from COCA COLA CO's 2023 Q2 report
[4] taken from the US GAAP taxonomy

and a namespace URI. The namespace URI is a URI that uniquely identifies the namespace and namespace prefix acts as a shorthand for the namespace.
For example the QName `us-gaap:Assets` identifies the element `Assets` in the namespace `us-gaap`.
In this example, the namespace prefix `us-gaap` is a shorthand for the namespace URI `https://xbrl.fasb.org/us-gaap/2022/elts/us-gaap-2022.xsd`, and together they form the namespace `us-gaap`.
QNames are used in the XBRL taxonomy to identify concepts, facts and other elements. Since they provide a robust and easy way to identify elements, we decided to use them in our API as well. However, there is one important difference between our QNames and the QNames used in the XBRL taxonomy: In the XBRL taxonomy, the mapping from namespace prefixes to namespace URIs depends on where the QName is used. In our API, there is a fixed, global mapping from namespace prefixes to namespace URIs.
Assume that the two followint XML schemas are referenced somewhere in the XBRL taxonomy. During the discovery phase of the DTS (Discoverable Taxonomy Set), the processor will download these two schemas and add them to the DTS. It will give both of them the same namespace prefix `us-gaap`, but different namespace URIs.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:us-gaap="https://xbrl.fasb.org/us-gaap/2021"
            targetNamespace="https://xbrl.fasb.org/us-gaap/2021"
            elementFormDefault="qualified">

  <xsd:import namespace="https://xbrl.fasb.org/us-gaap/2021" schemaLocation="

  <!-- Component and concept definitions ... -->

</xsd:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:us-gaap="https://xbrl.fasb.org/us-gaap/2022"
            targetNamespace="https://xbrl.fasb.org/us-gaap/2022"
            elementFormDefault="qualified">

  <xsd:import namespace="https://xbrl.fasb.org/us-gaap/2022" schemaLocation="

  <!-- Component and concept definitions ... -->

</xsd:schema>
```

Assume now that the user of the processor wants to access the concept `us-gaap:Assets`.
How does the processor know which version of the US-GAAP taxonomy to use?
There are two possible solutions to this problem:

1. The user has to specify not only the namespace prefix, but also the namespace URI.

   The problem with this solution is that the user has to know the namespace URI of the concept.

2. The processor completely ignores the namespace URI and always uses the namespace prefix to identify the concept.

The problem with this solution is that within the same document, different namespace prefixes can map to the same namespace URI.

This is the solution we chose for our processor.

To illustrate the problem with the second solution, consider the following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:us-gaap="https://xbrl.fasb.org/us-gaap/2021"
            xmlns:us-gaap1="https://xbrl.fasb.org/us-gaap/2021"
            targetNamespace="https://xbrl.fasb.org/us-gaap/2021"
            elementFormDefault="qualified">

    <xsd:import namespace="https://xbrl.fasb.org/us-gaap/2021" schemaLocati

    <!-- Component and concept definitions... -->

</xsd:schema>
```

In this example, the namespace URI `https://xbrl.fasb.org/us-gaap/2021` is associated with two different namespace prefixes. Once with the prefix `us-gaap` and once with the prefix `us-gaap1`.
When processing the XBRL instance, it is possible for the processor to associate a fact value with the concept `us-gaap1:Assets`. So if the user were to access the concept `us-gaap:Assets`, the processor would not find any facts associated with this concept.
This problem arises because the processor only compares the namespace prefixes and the local names, but ignores the namespace URIs.

3.5.1 Namespace normalization

Therefore our processor does a pre-processing step before processing the XBRL filing. We call this process *namespace normalization*. The purpose of namespace normalization is to ensure that the same namespace URI does not map to different prefixes.
The namespace normalization process works as follows:
The processor goes through both the instance and the DTS and finds all the prefix to namespace URI mappings. These mappings can be interpreted as a graph, where the nodes are the namespace prefixes/URIs and the edges are the mappings.
TODO: Make a picture of a nice bipartite graph with namespace prefixes and namespace URIs as nodes and mappings as edges.
The processor then finds all the connected components in this graph. Each connected component represents a set of namespace prefixes that map to the same namespace URI. Note that the prefixes might map to different versions of the same taxonomy.
For each connected component, the processor chooses one namespace prefix as the representative of the component. This prefix is the shortest prefix in the component. It then replaces all the other namespace prefixes in the component with the representative namespace prefix.
Furthermore, the processor finds one representative namespace URI for each connected component. This URI is the namespace URI of the representative namespace prefix. This representative namespace URI is the URI with the latest version of the taxonomy within the component. Whenever the user creates a QName e.g.

`us-gaap:Assets`, the processor associates `us-gaap` with the representative namespace URI.
The biggest advantage of namespace normalization is that it creates a flat prefix URI mapping that is consistent throughout the whole document. Whereas the namespace mapping in XML is hierarchical, the namespace mapping in our processor is flat. Therefore, the user does not have to worry about the namespace hierarchy.

3.5.2 Limitations of namespace normalization

Namespace normalization has some limitations. Not every namespace can be properly normalized.
The two problems that can arise are:

1. namespace normalization maps two namespaces into two, even though they should be merged into one.

2. namespace normalization maps two different namespaces into one, even though they should be kept separate.

For the first problem, consider the following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:us-gaap="https://xbrl.fasb.org/us-gaap/2021"
            xmlns:us-gaap1="https://xbrl.fasb.org/us-gaap/2022"
            targetNamespace="https://xbrl.fasb.org/us-gaap/2021"
            elementFormDefault="qualified">

    <xsd:import namespace="https://xbrl.fasb.org/us-gaap/2021" schemaLocation
    <xsd:import namespace="https://xbrl.fasb.org/us-gaap/2022" schemaLocation

    <!-- Component and concept definitions ... -->

</xsd:schema>
```

This example defines two different namespace prefixes, `us-gaap` and `us-gaap1`, that map to two different namespace URIs, `https://xbrl.fasb.org/us-gaap/2021` and `https://xbrl.fasb.org/us-gaap/2022`. When representing this mapping as a graph, we get the following graph:
TODO: Make a picture of a graph with two connected components.
This graph has two connected components, each with one namespace prefix and one namespace URI. A smart processor would realize that these two components are actually the same and would merge them into one component. Our processor, however, is will keep the two components separate and will not merge them.
For the second problem, consider the following schemas in a DTS:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:types="http://fasb.org/us-types/2023"
            targetNamespace="http://fasb.org/us-types/2023"
            elementFormDefault="qualified">

<xsd:import namespace="http://fasb.org/us-types/2023" schemaLocation="us-type

<!-- Component and concept definitions ... -->

</xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:types="http://www.xbrl.org/dtr/type/2020-01-21"
            targetNamespace="http://www.xbrl.org/dtr/type/2020-01-21"
            elementFormDefault="qualified">

<xsd:import namespace="http://www.xbrl.org/dtr/type/2020-01-21" schemaLoca

<!-- Component and concept definitions ... -->

</xsd:schema>
```

The first schema defines the namespace prefix `types` and maps it to the namespace URI `http://fasb.org/us-types/2023`. The second schema defines the namespace prefix `types` and maps it to the namespace URI `http://www.xbrl.org/dtr/type/2020-01-21`.
When representing this mapping as a graph, we get the following graph:
TODO: Make a picture of a graph with two connected components.
This graph has one connected component with one prefix and two namespace URIs.
A smart processor would realize that the two namespaces should be kept separate and would not merge them, since they associate the prefix `types` with two completely different namespaces.
Our processor, however, will merge the two namespaces into one, since they have the same prefix.

Appendix A

Irgendwas

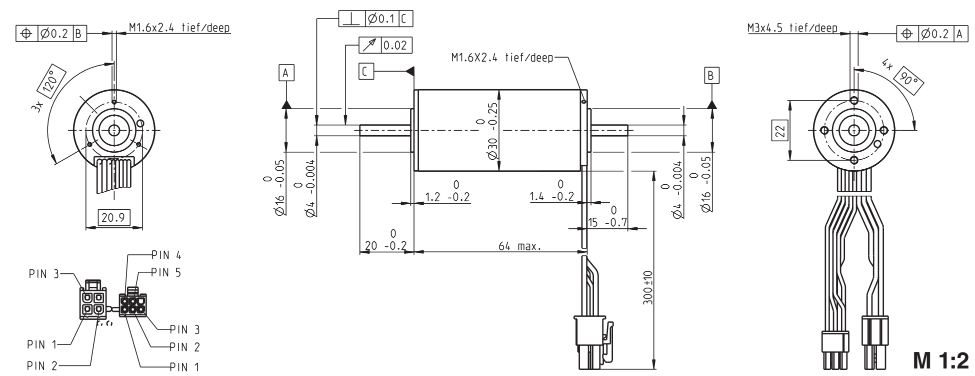
Bla bla . . .

Appendix B

Datasheets

EC-max 30 ∅30 mm, brushless, 60 Watt

maxon *EC*-max



M 1:2

- Stock program
- Standard program
- Special program (on request)

Part Numbers

Motor Data		272762	272763	272764	272765
Values at nominal voltage					
1 Nominal voltage	V	12	24	36	48
2 No load speed	rpm	7980	9340	9490	9350
3 No load current	mA	302	191	130	95.4
4 Nominal speed	rpm	6590	8040	8270	8130
5 Nominal torque (max. continuous torque)	mNm	63.6	60.7	63.7	64.1
6 Nominal current (max. continuous current)	A	4.72	2.66	1.88	1.4
7 Stall torque	mNm	381	458	522	519
8 Starting current	A	26.8	18.8	14.5	10.7
9 Max. efficiency	%	80	81	82	82
Characteristics					
10 Terminal resistance phase to phase	Ω	0.447	1.27	2.48	4.49
11 Terminal inductance phase to phase	mH	0.049	0.143	0.312	0.573
12 Torque constant	mNm/A	14.2	24.3	35.9	48.6
13 Speed constant	rpm/V	672	393	266	197
14 Speed/torque gradient	rpm/mNm	21.2	20.6	18.4	18.2
15 Mechanical time constant	ms	4.86	4.73	4.21	4.17
16 Rotor inertia	gcm²	21.9	21.9	21.9	21.9

Specifications

Thermal data

17	Thermal resistance housing-ambient	7.4 K/W
18	Thermal resistance winding-housing	0.5 K/W
19	Thermal time constant winding	2.76 s
20	Thermal time constant motor	1000 s
21	Ambient temperature	-40…+100°C
22	Max. permissible winding temperature	+155°C

Mechanical data (preloaded ball bearings)

23	Max. permissible speed	15000 rpm
24	Axial play at axial load < 6.0 N	0 mm
	> 6.0 N	0.14 mm
25	Radial play	preloaded
26	Max. axial load (dynamic)	5 N
27	Max. force for press fits (static)	98 N
	(static, shaft supported)	1300 N
28	Max. radial loading, 5 mm from flange	25 N

Other specifications

29	Number of pole pairs	1
30	Number of phases	3
31	Weight of motor	305 g

Values listed in the table are nominal.

Connection motor (Cable AWG 20)

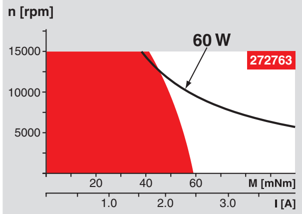
red	Motor winding 1	Pin 1
black	Motor winding 2	Pin 2
white	Motor winding 3	Pin 3
	N.C.	Pin 4
Connector	**Part number**	
Molex	39-01-2040	

Connection Sensors (Cable AWG 26)

yellow	Hall sensor 1	Pin 1
brown	Hall sensor 2	Pin 2
grey	Hall sensor 3	Pin 3
blue	GND	Pin 4
green	V_{Hall} 3…24 VDC	Pin 5
	N.C.	Pin 6
Connector	**Part number**	
Molex	430-25-0600	

Wiring diagram for Hall sensors see p. 35

Operating Range



Comments

Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

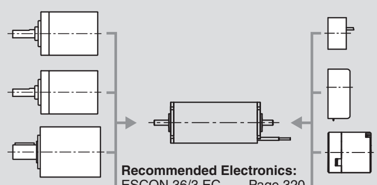
maxon Modular System

Overview on page 20 - 25

Planetary Gearhead
∅32 mm
8.0 Nm
Page 266

Koaxdrive
∅32 mm
1.0 - 4.5 Nm
Page 268

Planetary Gearhead
∅42 mm
3 - 15 Nm
Page 271



Encoder MR
500/1000 CPT,
3 channels
Page 302

Encoder HEDL 5540
500 CPT,
3 channels
Page 308

Brake AB 20
24 VDC
0.1 Nm
Page 346

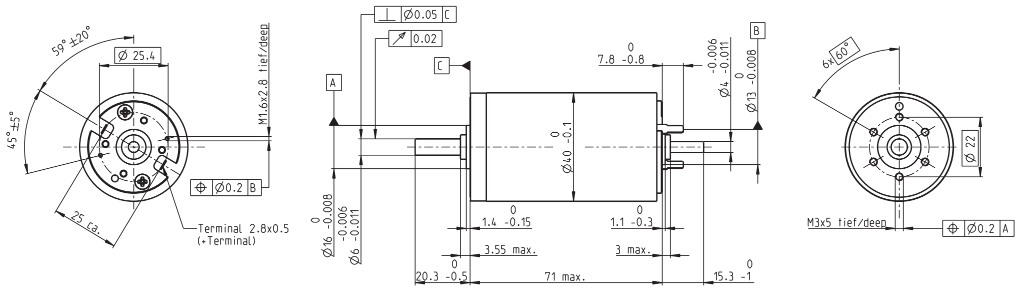
Recommended Electronics:

ESCON 36/3 EC	Page 320
ESCON 50/5, Module 50/5	321
ESCON 70/10	321
DECS 50/5	324
DEC Module 24/2	325
DEC Module 50/5	325
EPOS2 24/5, 50/5	331
EPOS2 P 24/5	334
EPOS3 70/10 EtherCAT	337
Notes	**24**

RE 40 ∅40 mm, Precious Metal Brushes, 25 Watt

NEW

maxon DC motor



M 1:2

Stock program
Standard program
Special program (on request)

Part Numbers

Motor Data		448588	448589	448590	448591	448592
Values at nominal voltage						
1 Nominal voltage	V	9	18	24	42	48
2 No load speed	rpm	2850	2850	2780	2920	2690
3 No load current	mA	49.7	24.8	18.1	11	8.62
4 Nominal speed	rpm	2610	2600	2480	2640	2410
5 Nominal torque (max. continuous torque)	mNm	87.8	87.8	88.2	87.6	87.6
6 Nominal current (max. continuous current)	A	2.96	1.48	1.09	0.65	0.524
7 Stall torque	mNm	873	956	794	895	818
8 Starting current	A	29	15.9	9.66	6.53	4.81
9 Max. efficiency	%	92	92	92	92	92
Characteristics						
10 Terminal resistance	Ω	0.311	1.14	2.49	6.43	9.97
11 Terminal inductance	mH	0.0824	0.33	0.613	1.7	2.62
12 Torque constant	mNm/A	30.2	60.3	82.2	137	170
13 Speed constant	rpm/V	317	158	116	69.7	56.2
14 Speed / torque gradient	rpm/mNm	3.27	2.98	3.51	3.27	3.3
15 Mechanical time constant	ms	4.85	4.29	4.36	4.14	4.13
16 Rotor inertia	gcm²	142	137	119	121	120

Specifications

Thermal data

17	Thermal resistance housing-ambient	4.65 K/W
18	Thermal resistance winding-housing	1.93 K/W
19	Thermal time constant winding	41.5 s
20	Thermal time constant motor	809 s
21	Ambient temperature	-20…+85°C
22	Max. permissible winding temperature	+100°C

Mechanical data (ball bearings)

23	Max. permissible speed	3330 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.025 mm
26	Max. axial load (dynamic)	5.6 N
27	Max. force for press fits (static)	110 N
	(static, shaft supported)	1200 N
28	Max. radial loading, 5 mm from flange	28 N

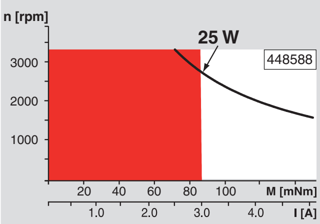
Other specifications

29	Number of pole pairs	1
30	Number of commutator segments	13
31	Weight of motor	480 g

Values listed in the table are nominal.
Explanation of the figures on page 71.

Option
Preloaded ball bearings

Operating Range



Comments

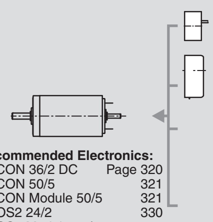
Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System

Overview on page 20 - 25



Encoder MR
256 - 1024 CPT,
3 channels
Page 303

Encoder HED_ 5540
500 CPT,
3 channels
Page 305/307

Recommended Electronics:

ESCON 36/2 DC	Page 320
ESCON 50/5	321
ESCON Module 50/5	321
EPOS2 24/2	330
EPOS2 Module 36/2	330
EPOS2 24/5	331
EPOS2 50/5	331
EPOS2 P 24/5	334
EPOS3 70/10 EtherCAT	337
Notes	**22**