

---

# Robowaifu Design Document

---

*The /robowaifu/ Community*  
*Chobitsu, general editor*  
*English translation*  
*version v200422*

## Abstract

This document describes the specification, design, engineering, implementation, capabilities, and usage of the *Robowaifu Reference Model A Series*. It is intended for designers, technicians, engineers and developers rather than for general lay audience consumption.

This manual is a living research and design document, and is in addition to the general *Robowaifu User's Guide*. It will be constantly changing and being updated for the foreseeable future. Each published revision will have a unique version number (based on date). The **/robowaifu/** community at large (as well as it's neighboring communities) contributed to the design ideas and specifications herein. Without them, this work would not have been possible.

## Contents

<b>I</b>	<b>Basic assumptions</b>	<b>8</b>
<b>1</b>	<b>Introduction</b>	<b>9</b>
1.1	Implementer requirements . . . . .	9
1.2	Engineering requirements . . . . .	9
1.3	Design requirements . . . . .	9
1.4	Sociological considerations . . . . .	9
<b>2</b>	<b>The art of robowaifu engineering &amp; design</b>	<b>10</b>
2.1	Common rules . . . . .	10
2.1.1	Technical design documents . . . . .	10
2.2	Special note for technical-description style documents . . .	11
<b>II</b>	<b>Physical considerations</b>	<b>12</b>
<b>3</b>	<b>Physical safety General</b>	<b>13</b>
3.1	Human safety . . . . .	13

3.2	Robowaifu safety . . . . .	13
3.3	Environment safety . . . . .	13
3.4	Detecting mechanical error-conditions . . . . .	13
<b>4</b>	<b>Mechanical</b>	<b>13</b>
4.1	Kinematic considerations . . . . .	13
4.1.1	Structural geometry considerations . . . . .	13
4.1.2	Mass is everything . . . . .	13
4.1.3	Thrown mass . . . . .	13
4.1.4	Inboard mass . . . . .	13
4.2	Armatures & frameworks . . . . .	13
4.2.1	Head . . . . .	13
4.2.1.1	Face . . . . .	13
4.2.1.1.1	Eyes & eyelids . . . . .	14
4.2.1.1.2	Eyebrows . . . . .	14
4.2.1.1.3	Mouth . . . . .	14
4.2.1.2	Neck . . . . .	14
4.2.2	Torso . . . . .	14
4.2.3	Arms . . . . .	14
4.2.3.1	Shoulder joints . . . . .	14
4.2.3.2	Elbow joints . . . . .	14
4.2.3.3	Wrist joints . . . . .	14
4.2.4	Hands . . . . .	14
4.2.4.1	Palm . . . . .	14
4.2.4.2	Thumb . . . . .	14
4.2.4.3	Fingers . . . . .	15
4.2.5	Legs . . . . .	15
4.2.5.1	Hip joints . . . . .	15
4.2.5.2	Knee joints . . . . .	15
4.2.5.3	Ankle joints . . . . .	15
4.2.6	Feet . . . . .	15
4.2.6.1	Toes . . . . .	15
4.3	Actuators, force-transfer, & control mechanisms . . . . .	15
4.3.1	Electrical motors . . . . .	15
4.3.2	Cable-based actuators . . . . .	15
4.3.2.1	Bowden cables . . . . .	15
4.3.2.2	Simple (pull) cables . . . . .	15
4.3.3	Screw (linear) actuators . . . . .	15

4.3.4	Lever (pushrod) actuators . . . . .	16
4.3.5	Pneumatic (artificial muscle) actuators . . . . .	16
4.3.6	Hydraulic actuators . . . . .	16
4.3.7	Unconventional actuators . . . . .	16
4.4	Lubrication & joints . . . . .	16
4.4.1	Fluid-distribution manifolds, pipes, & tubing . . .	16
4.4.2	Circulatory pumps & mechanisms . . . . .	16
4.4.3	Filtering & cooling . . . . .	16
4.4.4	Reservoirs . . . . .	16
4.4.5	Access & replenishment . . . . .	16
4.4.6	Joints servicing & replacement . . . . .	16
<b>5</b>	<b>External shell &amp; ‘skin’</b>	<b>16</b>
5.1	Hard-surface materials . . . . .	16
5.2	Soft-surface materials . . . . .	16
5.3	External shell & cooling-systems symbiosis . . . . .	16
<b>6</b>	<b>Cooling</b>	<b>17</b>
6.1	Passive cooling . . . . .	17
6.1.1	Materials . . . . .	17
6.2	Active cooling . . . . .	17
6.2.1	Fans . . . . .	17
6.2.2	Refrigeration . . . . .	17
6.3	Cooling-fluids & distribution . . . . .	17
6.3.1	Fluid-distribution manifolds, pipes, & tubing . . .	17
6.3.2	Circulatory pumps & mechanisms . . . . .	17
6.3.3	Reservoirs . . . . .	17
6.3.4	Access & replenishment . . . . .	17
<b>7</b>	<b>Power</b>	<b>17</b>
7.1	Power aquisition & production . . . . .	17
7.1.1	Household current . . . . .	17
7.1.2	RF induction . . . . .	17
7.1.3	Generative braking . . . . .	18
7.1.4	Solar trickle . . . . .	18
7.2	Power switching systems . . . . .	18
7.2.1	Automatic & failover switching . . . . .	18
7.2.2	Manual switching . . . . .	18

7.3	Power storage . . . . .	18
7.3.1	Batteries . . . . .	18
7.3.1.1	LiPo . . . . .	18
<b>8</b>	<b>Electrical, control, &amp; network wiring</b>	<b>18</b>
8.1	Electrical . . . . .	18
8.1.1	Electrical safety . . . . .	18
8.1.2	Electrical busses . . . . .	18
8.1.3	Electrical wiring . . . . .	18
8.2	Control . . . . .	18
8.3	Networking . . . . .	18
8.3.1	Network gear . . . . .	19
8.3.2	Network wiring . . . . .	19
8.3.3	Wireless networking . . . . .	19
<b>9</b>	<b>Computing resources</b>	<b>19</b>
9.1	Onboard computing resources . . . . .	19
9.1.1	Single-board computers (SBCs) . . . . .	19
9.1.1.1	Beaglebone Blue . . . . .	19
9.1.1.2	RaspberryPi & clones . . . . .	19
9.1.2	Microcontrollers . . . . .	19
9.1.2.1	Arduino Nano . . . . .	19
9.2	External computing resources . . . . .	19
9.2.1	Home servers & clusters . . . . .	19
9.2.2	Internet cloud providers . . . . .	19
<b>10</b>	<b>Sensors</b>	<b>19</b>
10.1	Visual . . . . .	20
10.1.1	Stereo & multipoint . . . . .	20
10.2	Audio . . . . .	20
10.2.1	Stereo & multipoint . . . . .	20
10.3	Tactile . . . . .	20
10.3.1	Response patches . . . . .	20
10.3.2	Overall system response . . . . .	20

<b>11 Mechanical construction approaches</b>	<b>20</b>
11.1 Safety . . . . .	20
11.2 Using the right tools . . . . .	20
11.3 Using construction rigs . . . . .	20
11.4 Manufacturing parts . . . . .	20
11.5 Fastening parts together . . . . .	20
11.6 Wiring parts together . . . . .	20
11.7 Stress & load-testing . . . . .	20
11.7.1 Testing individual components . . . . .	21
11.7.2 Testing subsystems . . . . .	21
11.7.3 Testing the overall system . . . . .	21
<b>III Software considerations</b>	<b>22</b>
<b>12 Software safety General</b>	<b>23</b>
12.1 Human safety . . . . .	23
12.2 Robowaiifu safety . . . . .	23
12.3 Environment safety . . . . .	23
12.4 Software error-detection & fault-tolerance . . . . .	23
<b>13 Software construction approaches</b>	<b>23</b>
13.1 The Python programming language . . . . .	23
13.2 The C++ programming language . . . . .	23
13.3 The C programming language . . . . .	23
13.4 IDEs . . . . .	23
13.5 Software-testing systems . . . . .	23
13.6 Debugging . . . . .	23
<b>14 Realtime performance considerations</b>	<b>23</b>
14.1 Realtime subsystems . . . . .	23
14.2 Non-realtime subsystems . . . . .	23
<b>15 IPCNet</b>	<b>24</b>
15.1 Introduction . . . . .	24
15.2 Data requirements . . . . .	25
15.3 Security . . . . .	26
15.4 Compromised systems . . . . .	27
15.5 Feature set . . . . .	27
15.6 Implementation . . . . .	27

15.7 Discussion, further Q&A . . . . .	28
<b>16 Control &amp; communications UI</b>	<b>29</b>
16.1 Onboard C&C UI . . . . .	29
16.2 External C&C UI . . . . .	29
<b>17 AI &amp; Machine Learning</b>	<b>29</b>
17.1 Foundations, briefly . . . . .	29
17.2 Current AI/ML frameworks . . . . .	29
17.2.1 TensorFlow . . . . .	29
17.2.2 PyTorch . . . . .	29
17.2.3 Keras . . . . .	29
17.3 Compute-resource considerations . . . . .	29
17.4 Our general robowaiifu ML approach . . . . .	29
17.4.1 Reinforcement Learning (RL) . . . . .	29
17.4.2 Deep Neural-Networks (DNNs) . . . . .	29
<b>IV Design considerations</b>	<b>30</b>
<b>V Reference, appendices, etc</b>	<b>31</b>
<b>18 How to change a L<sup>A</sup>T<sub>E</sub>X layout?</b>	<b>32</b>
18.1 Advantages and disadvantages of L <sup>A</sup> T <sub>E</sub> X . . . . .	32
18.2 Input files and class files . . . . .	32
18.3 Class files and packages . . . . .	32
18.4 Changing the layout, step by step . . . . .	33
18.4.1 Differences between desired and available layout . . . . .	33
18.4.2 Finding the original definition . . . . .	33
18.4.3 Writing a new package file . . . . .	34
18.4.4 Using the new package . . . . .	34
18.5 A simple example (Equation numbers) . . . . .	34
18.6 A more complex example (Reference Manual) . . . . .	35
18.6.1 Page layout . . . . .	35
18.6.2 Section headings . . . . .	36
18.6.3 Setting the margin notes . . . . .	37
18.6.4 Extensions . . . . .	37
<b>Appendix</b>	<b>38</b>

<b>A</b>	<b>The page structure in L<sup>A</sup>T<sub>E</sub>X</b>	<b>38</b>
<b>B</b>	<b>Description of the refman-class family</b>	<b>41</b>
B.1	Invocation . . . . .	41
B.2	Options . . . . .	41
B.3	Layout changes . . . . .	41
B.3.1	Page design . . . . .	41
B.3.2	Section headings . . . . .	42
B.3.3	Paragraphs . . . . .	42
B.4	Footnotes . . . . .	42
B.4.1	Description environment . . . . .	42
B.4.2	Positioning of margin notes . . . . .	42
B.4.3	Headers and Footers . . . . .	43
B.5	Additional commands . . . . .	43
B.5.1	Marginlabel . . . . .	43
B.5.2	Attention . . . . .	43
B.5.3	Seealso . . . . .	44
B.5.4	Maxipage environment . . . . .	44
B.5.5	Fullpage environment . . . . .	44
B.5.6	Noparskip . . . . .	44
B.5.7	Setleftmarginwidth . . . . .	44
B.5.8	Descriptioncolon . . . . .	44
B.5.9	Descriptionleft . . . . .	45
B.5.10	Maxipagerule . . . . .	45
B.5.11	Condbreak . . . . .	45
B.5.12	Example . . . . .	45
B.5.13	Pageperchapter . . . . .	45
B.5.14	Smallborder . . . . .	45
B.5.15	Dvips . . . . .	45
B.5.16	Pdftex . . . . .	45
B.5.17	Pagesize . . . . .	45
B.5.18	Ifpdfoutput . . . . .	46

---

## I. Basic assumptions

---

This part of the document is intended to be more or less read in order from start to finish. There is nothing complicated here, and information considered fundamental to the rest of the manual is included here.

Here we cover some of the important basics. Where does one even *begin* creating a robowaifu? Why, at the **beginning** of course!

# 1 Introduction

## 1.1 Implementer requirements

Every form of robowaifu that is created has at least two human roles involved: **Engineers**, and **Designers**. These may be the same man, but usually many men will be involved in teams, with roles often overlapping.

Engineer: The **engineering** will be crafted by the engineer.

*TBD.*

Designer: The **designs** will be crafted by the designer.

*TBD.*

## 1.2 Engineering requirements

When an engineer works to craft a robowaifu, numerous mechanical, electrical, computing, networking and other considerations must be taken into account.

1. The robowaifu engineering has to be defined. This is basically a job for a professional engineer, but for now Anon will have to step in.
2. The engineering has to be detailed with specifics inside engineering documents so it can be manufactured reliably.
3. *TBD.*

## 1.3 Design requirements

When a designer works to craft a robowaifu, numerous aesthetic, social and other considerations must be taken into account.

→ § 18 layout

1. The robowaifu design has to be defined. This is basically a job for a professional designer, but for now Anon will have to step in.
2. The design has to be detailed with specifics inside design documents so it can be manufactured reliably. § 18 contains more information about the software used to create these design documents.
3. *TBD.*

## 1.4 Sociological considerations

A **Robowaifu** has many sociological implications for her **Master**, and the designer and engineer must both take all these points into account when devising the functioning of the robowaifu's varied systems.

1. The social engagement for the master should be pleasing to him.
2. The social engagement for the master should be productive for him.
3. The social engagement for the master must not be harmful to him.<sup>1</sup>
4. *TBD.*

---

<sup>1</sup> This is not to say the robowaifu must never *admonish* her master when circumstances warrant it. For example, in matters of personal health & safety. To neglect to do so would in fact *be* harmful to her master. Just don't let her turn into some kind of nagging bitch. Therefore, some type of 'off' switch for this aspect is needed.

## 2 The art of robowaifu engineering & design

### 2.1 Common rules

There are few common rules because every kind of robowaifu has different requirements and meets a particular **Anon**'s need. Her design should therefore take into consideration *who* will own this specific model of robowaifu, and *why*.

An important criteria for consideration is if her master will live with the robowaifu with her serving him as a mere domestic (a catgirl meido), or if he wants to find a true partner with the robowaifu in 'marital' bliss (a true waifu) or some other variation (for example as just a simple chatbox).

In addition to that, the designer of the robowaifu has to take into consideration and accomodate certain other domestic conventions, like the living habits (sexual and otherwise) of her master. <sup>2</sup>

! → The main purpose of a robowaifu design is to ensure the master finds the satisfaction he wants with her, and to be able to enjoy and fully engage with his pretty robowaifu. However, for the design *documents*; structure, readability, and consistency is far more important than "being artsy". <sup>3</sup>

#### 2.1.1 Technical design documents

The following rules of thumb should be valid for most documents:

- Line spacing : The spacing between two lines should be larger than the spacing between two words to guide the eyes of the reader.
- Line length : The length of a line – or when using multicolumn layout of a column – should be about 60 characters. When lines get longer they are more difficult to read and it is easier to go to the wrong line after finishing the current one. Increasing the linespacing may help a little. On no account should code block lines exceed 80 characters. Conversely, when lines get too short it is difficult to set them as justified, and you will get lots of hyphenated words.
- Page layout : Normal text pages should look the same throughout the document. Figures, tables and special pages like the index need not appear in the same layout but should take as much space as needed.
- Margin notes : Margin notes are often more suitable than footnotes because they appear right next to the text they refer to. Special margin notes are the "attention sign" or the "dangerous bend" that guide the user to important parts of the text.
- Headings and Footings : Headings and footings should make it easier for the reader to orient himself in the document. If you expect readers to copy single pages from the document they should contain information about the paper as a whole, just in case you need more information or want to cite the whole paper.

If you expect the document to change often (like this one will), each page should contain version information or at least a date.

---

<sup>2</sup> Compare the lifestyles of different types of anons like "Outdoorsmen" and "Hikkis".

<sup>3</sup> Of course, the robowaifu *herself* should be beautiful, that's an important part of what makes her superior. But you, as a designer, should strive for engineering-grade quality in your documents, not fancifulness. We can save that for marketing.

## 2.2 Special note for technical-description style documents

Reference Manual Style : This L<sup>A</sup>T<sub>E</sub>X design style is suited for our purposes in general, and is used in this robowaifu design document. We recommend its use. <sup>4</sup>

- The text is printed in rather short lines in the right part of the page. This part is used for continuous reading.
- The wide left margin is used for headings and margin notes. Since you now have a wide margin it is easier to use long margin notes to supply additional information and to lead the reader to important parts of the documents. Please note that the margin is always on the left side thus two-sided printing does not look symmetrical. This is done on purpose, because the reader will always start reading at the left side, and with this layout section headers really “stand out”. In a symmetrical layout, half the headers would be buried in the text.
- Figures and tables are either inside the text column, inside the margin or, if necessary, fill the whole page.

→ § 18.6 refman ex.

§ 18.6 describes how to implement such a layout in L<sup>A</sup>T<sub>E</sub>X.

---

<sup>4</sup> The “PostScript Reference Manual” is another document that uses such a design.

---

## II. Physical considerations

---

Here we'll look at the physical, mechanical side of robowaifu engineering. This part contains mechanical and wiring diagrams, engineering specs, and other technical descriptions and documentation.

### **3 Physical safety General**

*TBD.*

#### **3.1 Human safety**

*TBD.*

#### **3.2 Robowaifu safety**

*TBD.*

#### **3.3 Environment safety**

*TBD.*

#### **3.4 Detecting mechanical error-conditions**

*TBD.*

### **4 Mechanical**

*TBD.*

#### **4.1 Kinematic considerations**

*TBD.*

##### 4.1.1 Structural geometry considerations

*TBD.*

##### 4.1.2 Mass is everything

*TBD.*

##### 4.1.3 Thrown mass

*TBD.*

##### 4.1.4 Inboard mass

*TBD.*

#### **4.2 Armatures & frameworks**

*TBD.*

##### 4.2.1 Head

*TBD.*

###### 4.2.1.1 Face

*TBD.*

4.2.1.1.1 Eyes & eyelids

*TBD.*

4.2.1.1.2 Eyebrows

*TBD.*

4.2.1.1.3 Mouth

*TBD.*

4.2.1.2 Neck

*TBD.*

4.2.2 Torso

*TBD.*

4.2.3 Arms

*TBD.*

4.2.3.1 Shoulder joints

*TBD.*

4.2.3.2 Elbow joints

*TBD.*

4.2.3.3 Wrist joints

*TBD.*

4.2.4 Hands

*TBD.*

4.2.4.1 Palm

*TBD.*

4.2.4.2 Thumb

*TBD.*

#### 4.2.4.3 Fingers

*TBD.*

### 4.2.5 Legs

*TBD.*

#### 4.2.5.1 Hip joints

*TBD.*

#### 4.2.5.2 Knee joints

*TBD.*

#### 4.2.5.3 Ankle joints

*TBD.*

### 4.2.6 Feet

*TBD.*

#### 4.2.6.1 Toes

*TBD.*

## 4.3 Actuators, force-transfer, & control mechanisms

*TBD.*

### 4.3.1 Electrical motors

*TBD.*

### 4.3.2 Cable-based actuators

*TBD.*

#### 4.3.2.1 Bowden cables

*TBD.*

#### 4.3.2.2 Simple (pull) cables

*TBD.*

### 4.3.3 Screw (linear) actuators

*TBD.*

#### 4.3.4 Lever (pushrod) actuators

*TBD.*

#### 4.3.5 Pneumatic (artificial muscle) actuators

*TBD.*

#### 4.3.6 Hydraulic actuators

*TBD.*

#### 4.3.7 Unconventional actuators

*TBD.*

### 4.4 Lubrication & joints

*TBD.*

#### 4.4.1 Fluid-distribution manifolds, pipes, & tubing

*TBD.*

#### 4.4.2 Circulatory pumps & mechanisms

*TBD.*

#### 4.4.3 Filtering & cooling

*TBD.*

#### 4.4.4 Reservoirs

*TBD.*

#### 4.4.5 Access & replenishment

*TBD.*

#### 4.4.6 Joints servicing & replacement

*TBD.*

## 5 External shell & ‘skin’

*TBD.*

### 5.1 Hard-surface materials

*TBD.*

### 5.2 Soft-surface materials

*TBD.*

### 5.3 External shell & cooling-systems symbiosis

*TBD.*

## **6 Cooling**

*TBD.*

### **6.1 Passive cooling**

*TBD.*

#### 6.1.1 Materials

*TBD.*

### **6.2 Active cooling**

*TBD.*

#### 6.2.1 Fans

*TBD.*

#### 6.2.2 Refrigeration

*TBD.*

### **6.3 Cooling-fluids & distribution**

*TBD.*

#### 6.3.1 Fluid-distribution manifolds, pipes, & tubing

*TBD.*

#### 6.3.2 Circulatory pumps & mechanisms

*TBD.*

#### 6.3.3 Reservoirs

*TBD.*

#### 6.3.4 Access & replenishment

*TBD.*

## **7 Power**

*TBD.*

### **7.1 Power aquisition & production**

*TBD.*

#### 7.1.1 Household current

*TBD.*

#### 7.1.2 RF induction

*TBD.*

### 7.1.3 Generative braking

*TBD.*

### 7.1.4 Solar trickle

*TBD.*

## 7.2 Power switching systems

*TBD.*

### 7.2.1 Automatic & failover switching

*TBD.*

### 7.2.2 Manual switching

*TBD.*

## 7.3 Power storage

*TBD.*

### 7.3.1 Batteries

*TBD.*

#### 7.3.1.1 LiPo

*TBD.*

## 8 Electrical, control, & network wiring

*TBD.*

### 8.1 Electrical

*TBD.*

#### 8.1.1 Electrical safety

*TBD.*

#### 8.1.2 Electrical busses

*TBD.*

#### 8.1.3 Electrical wiring

*TBD.*

### 8.2 Control

*TBD.*

### 8.3 Networking

*TBD.*

### 8.3.1 Network gear

*TBD.*

### 8.3.2 Network wiring

*TBD.*

### 8.3.3 Wireless networking

*TBD.*

## 9 Computing resources

*TBD.*

### 9.1 Onboard computing resources

*TBD.*

#### 9.1.1 Single-board computers (SBCs)

*TBD.*

##### 9.1.1.1 Beaglebone Blue

*TBD.*

##### 9.1.1.2 RaspberryPi & clones

*TBD.*

#### 9.1.2 Microcontrollers

*TBD.*

##### 9.1.2.1 Arduino Nano

*TBD.*

### 9.2 External computing resources

*TBD.*

#### 9.2.1 Home servers & clusters

*TBD.*

#### 9.2.2 Internet cloud providers

*TBD.*

## 10 Sensors

*TBD.*

## **10.1 Visual**

*TBD.*

### **10.1.1 Stereo & multipoint**

*TBD.*

## **10.2 Audio**

*TBD.*

### **10.2.1 Stereo & multipoint**

*TBD.*

## **10.3 Tactile**

*TBD.*

### **10.3.1 Response patches**

*TBD.*

### **10.3.2 Overall system response**

*TBD.*

## **11 Mechanical construction approaches**

*TBD.*

### **11.1 Safety**

*TBD.*

### **11.2 Using the right tools**

*TBD.*

### **11.3 Using construction rigs**

*TBD.*

### **11.4 Manufacturing parts**

*TBD.*

### **11.5 Fastening parts together**

*TBD.*

### **11.6 Wiring parts together**

*TBD.*

### **11.7 Stress & load-testing**

*TBD.*

11.7.1 Testing individual components

*TBD.*

11.7.2 Testing subsystems

*TBD.*

11.7.3 Testing the overall system

*TBD.*

---

### III. Software considerations

---

Here we'll look at the software side of robowaifu engineering.

This part contains code samples, diagrams, engineering specs, and other technical descriptions and documentation.

## 12 Software safety General

*TBD.*

### 12.1 Human safety

*TBD.*

### 12.2 Robowaifu safety

*TBD.*

### 12.3 Environment safety

*TBD.*

### 12.4 Software error-detection & fault-tolerance

*TBD.*

## 13 Software construction approaches

*TBD.*

### 13.1 The Python programming language

*TBD.*

### 13.2 The C++ programming language

*TBD.*

### 13.3 The C programming language

*TBD.*

### 13.4 IDEs

*TBD.*

### 13.5 Software-testing systems

*TBD.*

### 13.6 Debugging

*TBD.*

## 14 Realtime performance considerations

*TBD.*

### 14.1 Realtime subsystems

*TBD.*

### 14.2 Non-realtime subsystems

*TBD.*

## 15 IPCNet

Inter-Process Communication Network Library.

*TBD.*

### 15.1 Introduction

Q&A :

- What is the project? IPCNet is a proposed parallel computing library that provides an easy-to-use interface for processes implemented in different languages across many platforms to share their data and provide data services to each other.
- Why create this project? Developers require a way to focus on building projects they find interesting and affordable to make, while others can quickly drop these components into their own robowaifu project.
- What do processes control? Processes have control over their data and subroutines and define the permissions and rules which they may be accessed, modified and executed. Processes may implement their own whitelists and blacklists, rate limits and usage limits. When processes query or request data it is just that, a request. Other processes do not have to fulfill it.
- What is it intended for? IPCNet is intended for parallel processes working together in collaboration, such as the components of a robot plugged into each other. IPCNet is not intended for programs with tightly coupled dependencies. When a process requests data there may be multiple processes available to fulfill the request.
- What is the main focus? The main focus is to connect developers' projects together and unite them with a single interface.
- What's different? IPC libraries that exist are specific to one or two programming languages and are not easy for developers to quickly include into their projects and interface processes across different languages or even different platforms. Parallel computing and grid computing software is often tooled for special purposes and are not intended for general use either. IPCNet provides a way for all kinds of processes and systems to interact, safely and quickly.

Development Philosophy :

- Simple design IPCNet should be simple in its implementation and its interface. The interface should work the same across all platforms and programming languages and parse communicated data into a proper format that the receiving process can readily use. The library and processes that use it should not obfuscate their subroutines beyond what is sensible for security and failsafety.

*“Relying on complex tools to manage and build your system is going to hurt the end users. [...] If you try to hide the complexity of the system, you'll end up with a more complex system. Layers of abstraction that serve to hide internals are never a good thing. Instead, the internals should be designed in a way such that they NEED no hiding.” — Aaron Griffin*

Robust and fail-safe IPCNet needs to be robust, fault tolerant and fail-safe. Data coming in through various input methods may contain errors from interference or malice and need to be corrected or discarded. Errors and failures should be handled gracefully and degrade the system in a way that will cause minimal harm to the equipment, other equipment, to the environment and to people. The library and processes implementing the library should consider the following questions:

1. How critical is the component/process?
2. How likely is the component/process to fail?
3. How can the component/process be made fault tolerant?
4. How can the component/process be designed to reduce impact and damage to the rest of the system in the event of failure?

*TBD.*

## 15.2 Data requirements

*TBD.*

Data Transfer : All forms of data should be transmittable over the network with low-latency and high throughput. Some forms of data that need careful consideration:

- Function parameters passed on the call-stack
- General data structures, such as stack and heap C-arrays
- AI tensor data (different libraries need to be able to exchange tensor data)
- Streaming video, raw and encoded
- Streaming audio, raw and encoded
- Other sensor data, such as haptic sensors, SONAR, LIDAR &tc.
- General files

Beyond function parameters, data types will need to be properly annotated with further metadata, describing the encoding of the data so it can be properly decoded and read by other processes.

Data validation : All data transmitted over the network needs to be checked for integrity. It must be free from errors, both ones caused by potential interference sources, and ones devised through malicious intent. It is expected that ongoing pentesting will be a normal, regular part of the robowaiifu engineering processes.

Data should be traceable to its source and (optionally) cryptographically tagged historically, indicating which processes/devices modified it, and in what order, so that data pipelines can be quickly inspected and debugged. This approach needs to function under normal runtime conditions to allow for realistic validations.

Modularity : Processes connected together through IPCNet should be easily separated and recombined, with the benefit of flexibility and variety of

choice in data and service providers. However, in the spirit of simple design these processes should not hide their complexity behind the interface beyond what is reasonable for security and safety, to ensure the reusability and versatility of these programs.

- Fault tolerance :
- No single point of failure – If a system experiences a failure, it must continue to operate without interruption during the repair process.
  - Fault isolation to the failing component – When a failure occurs, the system must be able to isolate the failure to the offending component.
  - Fault containment to prevent propagation of the failure – Some failure mechanisms can cause a system to fail by propagating the failure to the rest of the system. Mechanisms that can isolate a rogue component or failing component when it is unable to contain a fault are required to protect the system.
  - Availability of reversion modes – the abandonment of one or more recent changes in favor of a return to a previous version. If a component update breaks the system, it should revert back to the last working version.

### 15.3 Security

If a malicious entity successfully alters the data of an IPC socket, it/they are effectively tampering with a low-level mechanism part of the core operating system. If an attacker is able to do this—or even if they just manage to intercept the internal data—then that device has already been compromised.

Any processes with direct access to the Internet could potentially be exploited to create malicious requests within the IPC network or even become fully compromised by arbitrary code execution. Direct connection of the robowaifu with the Internet—or bridged connections out the Internet—is therefore highly discouraged. Use air-gapped systems at the least. Strongly validated data sources, &tc. are instead encouraged.

Additionally, any process could connect to the IPCNet and send malicious requests not created through the IPCNet library. Potentially rogue processes need to be identified, mitigated and isolated. Inputs from requests need to be sanitized and bad requests refused.

- Permissions :
- Process routines registered on the network should have explicitly defined read, write and execution permissions for processes and process groups.

*TBD.*

- Process Groups :
- Processes can register processes into groups and give them their own permissions.

*TBD.*

- Denial of Service mitigation :
- TBD.*

- Identify normal conditions for network traffic
- Filter malicious traffic (connection tracking, component reputation lists, black/whitelisting, or rate limiting)

*TBD.*

#### 15.4 Compromised systems

If a robowaifu subsystem is compromised by an attacker there is no reasonable way to verify the trustworthiness of processes thereafter. Data may be modified or processes abruptly shutdown by an attacker with intent to cause damage or harm.

This effectively implies the entire robowaifu system is compromised as well. Automatic checks (and backups) should be in place to detect compromised states and immediately shutdown gracefully & safely. Diagnostic information similar to the infamous BSOD must be made available to the robowaifu's Master to assist with the technical remediation efforts.

*TBD.*

#### 15.5 Feature set

General features : *TBD.*

- P2P networking for processes – connected processes form a mesh network and can communicate with each other even if they're not directly connected.
- Announce data – processes can announce to the network their data and data services.
- Find data and data services – processes can search the network for data and other processes providing data services.
- Query data – processes can query data that other processes already have in memory or on disk and have made available.
- Subscribe to data and data services – processes can subscribe to data and receive new updates whenever it is changed, or subscribe to services that provide a stream of data, continuously or at regular or irregular intervals.
- Offer service – processes can provide data services to other processes to execute and produce new data on request. These services may be public or private.
- Request service – processes can request data providers to execute subroutines that produce new data.

#### 15.6 Implementation

*TBD.*

Booting :

Event-driven interface :

Functions & returns :

Arguments & parameters :

Metadata :

IPC sockets :

Network sockets :

Security :

- Status Codes :
- Success
    - 100 OK
    - 101 Created
    - 102 Accepted
    - 103 No Data
    - 104 Partial Data
  - Client errors
    - 200 Bad Request
    - 201 Unauthorized
    - 202 Forbidden
    - 203 Not Found
    - 204 Method Not Allowed
    - 205 Too Many Requests
    - 206 Upgrade Required
  - Service errors
    - 300 Internal Process Error
    - 301 Not Implemented
    - 302 Bad Gateway
    - 303 Service Unavailable
    - 304 Gateway Timeout
    - 305 IPCNet Version Not Supported
    - 306 Insufficient Storage
    - 307 Insufficient Memory
    - 308 Loop Detected
    - 309 Network Authentication Required

## 15.7 Discussion, further Q&A

*TBD.*

1. How can we gracefully shutdown the system? A robot may need to move to a safe location and shutdown in a way that does not cause damage to the components, other equipment near it, the environment or nearby people.
2. How can we gracefully shutdown a compromised system? An attacker who has compromised a device in the network may modify data or shut it down with intent to cause damage or harm.
3. How critical is the component/process?
4. How likely is the component/process to fail?
5. How can the component/process be made fault tolerant?
6. How can the component/process be designed to reduce impact and damage to the rest of the system in the event of failure?

7. How can we gracefully shutdown the system?
8. How can we gracefully shutdown a *compromised* system?

## **16 Control & communications UI**

*TBD.*

### **16.1 Onboard C&C UI**

*TBD.*

### **16.2 External C&C UI**

*TBD.*

## **17 AI & Machine Learning**

*TBD.*

### **17.1 Foundations, briefly**

*TBD.*

### **17.2 Current AI/ML frameworks**

*TBD.*

#### 17.2.1 TensorFlow

*TBD.*

#### 17.2.2 PyTorch

*TBD.*

#### 17.2.3 Keras

*TBD.*

### **17.3 Compute-resource considerations**

*TBD.*

### **17.4 Our general robowaifu ML approach**

*TBD.*

#### 17.4.1 Reinforcement Learning (RL)

*TBD.*

#### 17.4.2 Deep Neural-Networks (DNNs)

*TBD.*

---

## IV. Design considerations

---

Here we'll look into some of the aesthetic and other aspects of robowaifu designs.

This part contains design diagrams, design specs, and other design descriptions and documentation.

---

## V. Reference, appendices, etc

---

This part is a grab-bag of general reference information.

## 18 How to change a L<sup>A</sup>T<sub>E</sub>X layout?

### 18.1 Advantages and disadvantages of L<sup>A</sup>T<sub>E</sub>X

Advantages: The big advantage of L<sup>A</sup>T<sub>E</sub>X is, that it implements a “generic” or “logical” design. This means that the author has to specify the *meaning* of special parts of the text like: headings, citations, lists, literature references, and so on. These logical definitions will be processed by the system and printed in the “correct” way. The meaning of *correct* is defined in the document class and additional packages.

The opposite of this is the “visual” design that most text processors use. Here the author has to know the correct way to set certain parts of the text and take care of the correct printing.

The logical design makes it easier for the author to write consistent documents (i. e., same font and fontsize for section headings of the same level, same layout for lists and enumerations, ...).

L<sup>A</sup>T<sub>E</sub>X is powerful and flexible: you can define arbitrary document design by changing the definitions in the class files or overwrite them with packages. This is easier than you probably think.

Disadvantages: The main disadvantage of L<sup>A</sup>T<sub>E</sub>X was that the author originally had only limited means to change the layout and that he had only four classes to choose from. This changed a great deal with the improved Koma-Script classes for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> and today it’s easy to use with a small learning curve.

### 18.2 Input files and class files

According to the principle of separation of content and design, there are two kind of files:

- The content and the logical structure of a document are defined in the L<sup>A</sup>T<sub>E</sub>X input file.
- The design (layout) is defined in the class files and packages.

Which class and packages files a document will use is defined at the beginning of the input file. The `\documentclass` command selects the class and the `\usepackage` command specifies additional packages.

To generate a document you need at least two files, the input file and a class file.

- Similar documents (that appear in a series) have the same layout because the layout is defined in a file of its own and not part of the document (this is similar in concept to the way CSS works).
- You can output the same content without much rework using different layouts, e. g., as an article for a magazine and as a chapter for a dissertation.

### 18.3 Class files and packages

L<sup>A</sup>T<sub>E</sub>X supports a hierarchy of layout definitions.

- The first file processed is the class file that is specified inside the curly braces of the `\documentclass` command. This defines the kind of document you want to write.

- The optional argument of the `\documentclass` command inside the square brackets defines class options which select variants of the basic layout, such as different font sizes.
- The last step is reading the packages specified by the `\usepackage` command. This command again takes options to select the layout.
- You can change layout parameters in the input file, but this is discouraged because it violates the principle of separation of content and design.

There are some important differences between class and package files and “normal” input files:

- Class and package files should only contain definitions. They must not output text.
- The “at”-sign `@` is treated as a letter and therefore may appear in command names. Most internal commands of  $\LaTeX$  contain an `@` to prevent the author from using them accidentally.
- The extension of the file is `.cls`, `.clo` or `.sty` instead of `.tex`.

## 18.4 Changing the layout, step by step

! :-) First off Anon, it’s much simpler to begin with an already-existing document that is close to your needs (e. g., just making a copy of and modifying this robowaifu design document would be a good place to start, and then you can basically just skip the rest of this section).

*-It’s how I did it, using the [refman](#)  $\LaTeX$  package. -Chobitsu*

It also is usually easier to change existing class files instead of writing a new one from scratch. In many cases it is even sufficient so replace some definitions and put them into a package instead of creating a new class.

Please note that you are *not* allowed to change the standard classes distributed with  $\LaTeX$ . You *have* to change the name when you want to make changes. That is another reason to put small changes in packages.

### 18.4.1 Differences between desired and available layout

The first step is to determine the difference between the layout you have and the layout you want.

### 18.4.2 Finding the original definition

The next step is to find out where the original layout is defined. It is best to search the files in the following order:

1. the  $\LaTeX$  manual by Leslie Lamport,
2. the  $\LaTeX$  documentations files `*.dtx` for the classes or packages
3. the  $\LaTeX$  documentations files `*.dtx` for the kernel,
4. the  $\TeX$ book by Donald E. Knuth.

The files are usually documented quite well so you should be able to change things even if you don’t understand everything.

### 18.4.3 Writing a new package file

The third step is to create a new package. You choose an appropriate name for the package (like `mysty`) and create a filename by adding the extension `.sty`.

This file will only contain the definitions you want to change or the new commands you want to define.

If you want to change definitions or certain parameters, the best way is to copy them from the original file and modify them according to your liking.

Defining new commands is easier when you find similar commands in the original files which you can change.

It is always a good idea to include the reason you wrote the package, the changes it makes and the new commands it defines in the file. You should include the date of the last change and the  $\text{\LaTeX}$  version it works with, just in case some internal  $\text{\LaTeX}$  commands you use will change.

When writing larger packages, it is an even better idea to use the `docstrip` program which is used to document the  $\text{\LaTeX} 2\epsilon$  files. Thus you have your code and documentation in one file and it's easier to keep them from going out of sync.

### 18.4.4 Using the new package

To use the new package, you call it with the `\usepackage` command. This command executes the code of your package and changes the layout as desired.

Example:

```
\documentclass[11pt,twoside,a4paper]{article}
\usepackage{mysty} %<- This calls the package "mysty"
```

You shouldn't need to change anything else in your input file, unless you defined new commands or environments that are not available in standard  $\text{\LaTeX}$ .

! → When you copy your input file to a different computer you have to include your new packages as well. Otherwise the document can't be processed.

## 18.5 A simple example (Equation numbers)

Let's assume that you want to write an article where the equations are numbered separately in every section. In the  $\text{\LaTeX}$  manual you find a notice that the `report` class does something similar for every chapter.

Looking into the file `report.cls` you will find the following commands that deal with equation numbers:

```
\@addtoreset{equation}{chapter}
\def\theequation{\thechapter.\arabic{equation}}
% or in LaTeX2e since 1995/06/01:
\renewcommand\theequation{
    \thechapter.\@arabic\c@equation
}
```

You don't necessarily need to understand these two commands in detail.

Now you create an new file with the name `eqpersec.sty`<sup>5</sup> and copy the commands above into that file. After that you replace every occurrence of `chapter` with `section` and add some comments.

```
% This is equation_per_section.sty
% Short name: eqpersec.sty
% Original file by Hubert Partl 1988
% Modified by Axel Kielhorn 1996/01/01
% to support LaTeX 1995/06/01 and later
%
% reset the equation counter at the start
% of a new section
%
\@addtoreset{equation}{section}
% Equationnumber = sectionnumber.equationnumber
% Use only one of the below
% depending on you LaTeX version
%
%\def\theequation{\thesection.\arabic{equation}}
% or in more recent versions of LaTeX
\renewcommand\theequation{
    \thesection.\@arabic\c@equation
}
```

Whenever you use a `\usepackage{eqpersec}` command as in

```
\documentclass[11pt]{article}
\usepackage{eqpersec}
```

you will get equations numbered according to your conventions.

## 18.6 A more complex example (Reference Manual)

We want to create a layout similar to the one used in the *PostScript Reference Manual*, with a wide left margin for headings and margin notes and a small margin at the right and bottom.

### 18.6.1 Page layout

To define the new layout we use the commands described in the  $\LaTeX$  manual. For full details see the file `refman.dtx`.

Horizontal: First we define two new names for length that we will use often:

`\fullwidth` is the width of the whole page minus a margin of 1 inch on every side.

$$\text{fullwidth} = \text{paperwidth} - 2 \text{ inch}$$

From this the width of the text is calculated.

$$\text{textwidth} = \text{fullwidth} \times \text{textfraction}$$

`\leftmarginwidth` is the width of the left margin that will be used for headings and margin notes.

$$\text{leftmarginwidth} = \text{fullwidth} - \text{textwidth}$$

<sup>5</sup> Depending on the computer you are using the name may be different like `EQPERSEC.STY` on a CYBER running NOS/VE. But note that you must not use spaces in the filename.

This is a little more difficult in reality because the lengths have to be rounded to full points and a possible two column layout – as used in the index – must be taken into consideration.

Vertical: The vertical layout is a little more difficult because you have to deal with the page header and footer.

$$\text{textheight} = \text{paperheight} - 2.5 \text{ inch}$$

The result of this calculation is rounded to full lines. Depending on the page style – headings or footings – it is shifted up or down by one line.

## 18.6.2 Section headings

The headings have to be modified to make them extend into the left margin.

In file `classes.dtx` we find the `\@startsection` command that defines the layout of the headings. Only parameters 4 to 6 are relevant for us: parameter 4 is the space above and parameter 5 the space below the section. The 6th parameter does the actual formatting.

This is the original definition:

```
\newcommand\section{\@startsection
  {section}{1}{\z@}%
  {-3.5ex plus -1ex minus -.2ex}%
  {2.3ex plus .2ex}%
  {\normalfont\Large\bfseries}}
```

The commands for sub- and subsubsections are similar. Note that the measures are all in `ex`, thus depending on the font size used.

We define a new command `\secshape` to format the headings. This command uses the whole width of the page for the heading. To discourage hyphenation of the heading we give it a high penalty. This still allows hyphenation when absolutely necessary.

```
\newcommand\secshape{%
  \leftskip=-\leftmarginwidth%
  \rightskip=\@flushglue%
  \hyphenpenalty=2000}
```

This command is inserted into the 6th parameter of `\@startsection`.

Since the headings now extend into the left margin, we can use a smaller font and reduce the space between the text and the heading. The new definition looks like the following:

```
\newcommand\section{\@startsection
  {section}{1}{\z@}%
  {-2ex plus -1ex minus -.2ex}%
  {0.5ex plus .2ex}%
  {\secshape\normalfont\large\bfseries}}
```

### 18.6.3 Setting the margin notes

The margin notes should always appear on the left side of the text. The normal layout puts them into the outer margin in twoside layout.

The file `latex.dtx` contains the definition of the `\@addmarginpar` command which is responsible for the margin notes. We don't have to understand the whole definition; the important part is the internal variable `\@tempcnta` that is either `\@ne` (1) when the note should appear on the right side of the text or `\m@ne` (-1) when it should appear on the left side.

This is done by the following lines:

```
\@tempcnta\@ne
\if@twocolumn
  \if@firstcolumn \@tempcnta\m@ne \fi
\else
  \if@mparswitch
    \ifodd\c@page \else\@tempcnta\m@ne \fi
  \fi
  \if@reversemargin \@tempcnta -\@tempcnta \fi
\fi
```

which we simply replace by:

```
\@tempcnta\m@ne
```

The remaining lines that handle the setting of the margin note depending on the parameter `\@tempcnta` are left unchanged.

### 18.6.4 Extensions

The definitions described above are sufficient for simple applications but in practical use one may want some additional commands.

→ Appendix B

You will find the description for the whole new class in the appendix B.

## Appendix

### A The page structure in $\text{\LaTeX}$

→ Fig. 1

This appendix describes how the actual page is build from its components and how they are influenced by  $\text{\TeX}$ 's parameters. (Figure 1 has been created by Nelson Beebe at the University of Utah.)

- text area : The normal text area (“Body”) contains the running text including footnotes, tables and figures. The headings, footer and margin notes do *not* belong to the text area.
- The text area has the width `\textwidth` and the height `\textheight`.
- In a two column layout the text area is split into two columns, with the width `\columnwidth` each and a space of `\columnsep` between them. Thus the `\columnwidth` is a little bit smaller than half the `\textwidth`.
- `\textwidth` and `\columnwidth` should be a multiple of the width of one character in the `tt` font.
- `\textheight` should be multiple of the line height `\baselineskip`, increased by the constant value of `\topskip`.
- Indentations inside the text area are defined with `\leftskip` and `\rightskip`. These parameters should not be changed explicitly by the user but rather implicitly through environments.
- left margin : The left margin is either `\odd-` or `\evensidemargin` plus 1 inch. Both parameters have the same value, unless the `twoside` option is given.
- top margin : The top margin is the sum of `\topmargin`, `\headheight` and `\headsep` plus 1 inch.
- right margin : The right margin is the paper width minus the left margin and the text area.
- bottom margin : The bottom margin is the paper height minus the top margin and the text area.
- heading : The heading is inside the top margin with a space of `\headsep` between the lower border of the header and the upper border of the text area. Above the header is a free space of `\topmargin` increased by 1 inch.
- footing : The footer is inside the bottom margin with a space of `\footskip` between the lower border of the text area and the lower border of the footer.
- margin notes : Margin notes are inside the left or right margin. They have a width of `\marginparwidth` and a space of `\marginparsep` between the margin note and the text area. The vertical space between two margin notes is `\marginparpush`.

The `paperheight` consists of the following elements (from top to bottom):

1 inch  
`\topmargin`  
`\headheight`  
`\headsep`  
`\textheight`  
`\footskip`  
remaining page.

On pages with margin notes in the right margin the `paperwidth` consists of the following elements:

1 inch  
`\oddsidemargin` or `\evensidemargin`  
`\textwidth`  
`\marginparsep`  
`\marginparwidth`  
remaining page

With the option `twoside` the left pages change to

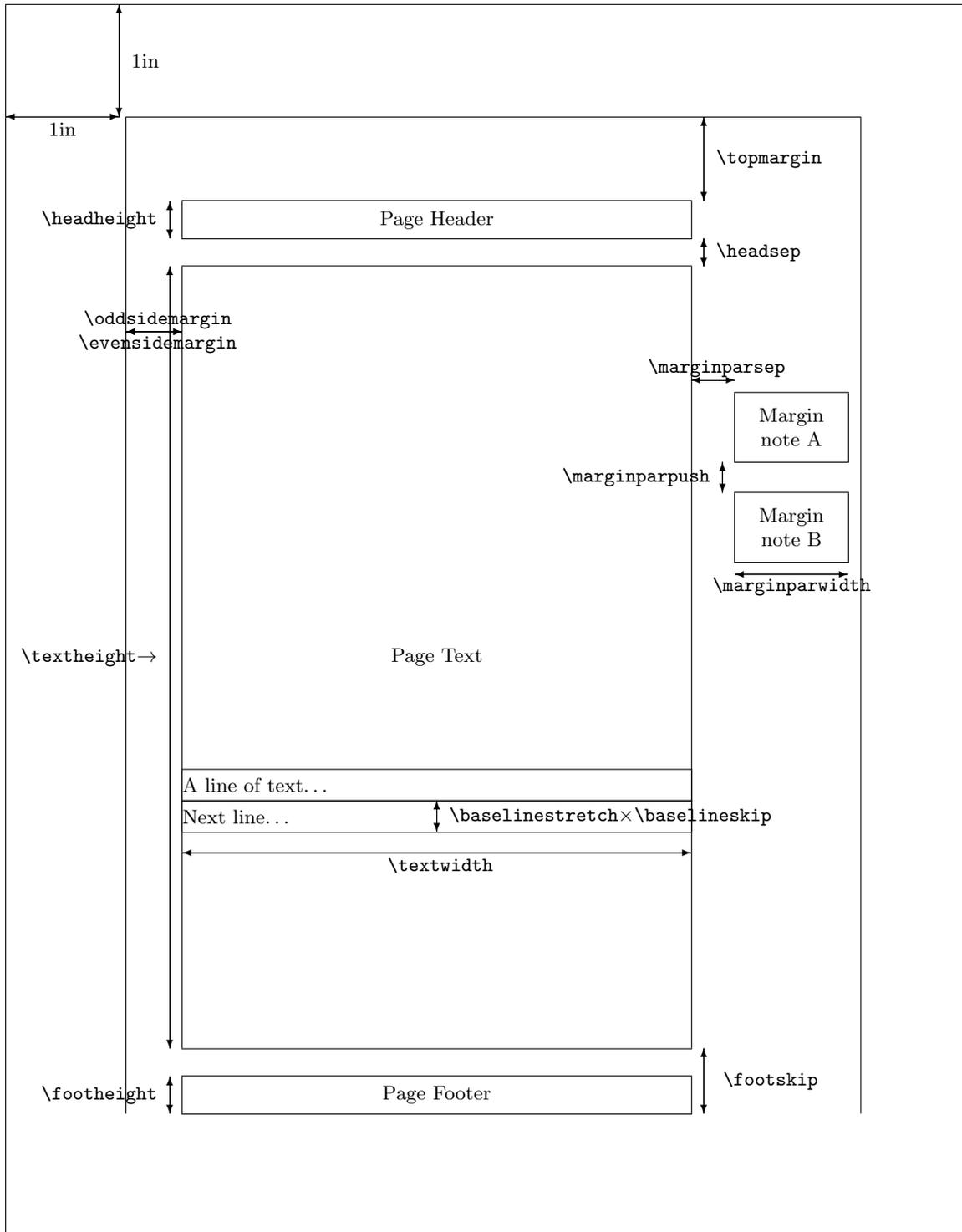
1 inch  
`\evensidemargin`  
`\textwidth`  
remaining page

Comments: The parameters `\topmargin`, `\oddsidemargin`, and `\evensidemargin` may be negative. In this case, the margin will be smaller than 1 inch. The same is true for `\leftskip` and `\rightskip` which leads to text that is wider than the text area.

Extensive treatment and figures to this subject may be found in the TUGBOAT Vol.9, No.1 (April 1988).

The parameter `\footheight` is no longer defined in L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> since no-one used it.

Figure 1: Page layout



## B Description of the refman-class family

The `refman.sty` was defined at the EDV-Zentrum (computing center) of the TU<sup>6</sup> Wien. This layout is suitable for reference manuals, technical descriptions and similar applications. It is based on the ideas shown in previous sections: The layout has a wide left margin for headings and margin notes and smaller margins on the right side, the top and the bottom.

In 1994 this layout was re-implemented as a class for the new L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>. This made it possible to include some minor improvements, such as the support of different paper sizes. The `refman.sty` was split into two classes `refrep`, similar to `report` and `refart`, similar to `article`. These classes differ in the layout of the header and footer. The `refart` does not support the `\chapter` command.

The current version of both classes is described in this document. It serves as an example for the layout.

### B.1 Invocation

The L<sup>A</sup>T<sub>E</sub>X local guide (if available) shows if this class is available at your T<sub>E</sub>X installation or where to install it. To use the `refart` class, simply call it with the `\documentclass` command:

```
\documentclass[11pt,a4paper]{refart}
\usepackage{german} % other packages you may want
```

### B.2 Options

The `refart` class replaces `article` and `refrep` replaces `report`. They support all options of these classes except for the `twocolumn` option.

It supports the additional option `square` which makes the `\textheight` equal to the `\textwidth`.

Neither `refart` nor `refrep` support two column layout, thus the commands `\twocolumn` and `\onecolumn` must not be used.

The index will be set in two column format and you can't change it with the means of this class.

### B.3 Layout changes

#### B.3.1 Page design

Horizontal: In this design the usable area for text (`\fullwidth`) is calculated as the paper width minus 2 `\papermarginwidth`. The default value for `\papermarginwidth` is 1 Inch.

The option `smallborder` reduces `\papermarginwidth` to 0.25 Inch. This is more suitable for documents viewed on screen, especially when combined with the `a5paper` and `landscape` options.

Only a fraction of this width is used for the running text (`\textwidth`), the remaining part forms a wide left margin (`\leftmarginwidth`) which is used for headings and margin notes. The `\textwidth` is 70 % of the `\fullwidth` by default, but this can be changed with

---

<sup>6</sup> Technical University

the `\setttextfraction` command which accepts arguments between 0 and 1.

Vertical: The text height is calculated as the paper height minus  $2 \times \text{\papermarginwidth}$ . The `\topmargin` is modified by some pagestyles. (see B.4.3).

The pages are always set with a ragged bottom.

### B.3.2 Section headings

The headings for `\section`, `\subsection`, and `\subsubsection` extend into the left margin, thus using the full width of the page. They are not justified and hyphenation is discouraged. A small space is kept free above and below the heading. Headings for `\section` and `\subsection` are set in a bold font.

The `refrep` class defines a different layout for the `\chapter` command: It always starts a new page and prints the chapter headings in a large bold font with a thick line above and below. This heading uses the full width of the page.

A similar heading is created by the `\part` commands which is available in both classes. It uses a roman part number instead of the arabic section number.

The `\maketitle` commands sets the title of the document in the same layout when no special title page is requested. (This is the default for `refart`. To suppress the title page in a `refrep` document, you can use the `notitlepage` option.) The name of the author and the date is printed in italic flush right below the document title.

### B.3.3 Paragraphs

Paragraphs are separated by a vertical space (`\parskip`) of half a line ( $0.5 \times \text{\baselineskip}$ ) plus a stretchable length of 2 pt. Paragraphs are not indented.

The vertical spacing inside, above and below a list environment is the same as in the running text.

## B.4 Footnotes

The footnote layout consists of a small margin (1em) which contains the footnote symbol. A small space is set between the symbol and the footnote text. The paragraphs of the footnote are not indented. There is currently no space between two footnotes, I'm not sure if this will stay this way. The footnote symbol is set as a superscript. This may change in later versions. I'm relying on user feedback to finally solve this.

### B.4.1 Description environment

The `description` environment will use the whole left margin for the description label.

→ Section 18

You will find examples in the section 18.

### B.4.2 Positioning of margin notes

Margin notes (`\marginpar`) are always put into the left margin. They use the whole width of the margin.

The minimum space between two margin notes is set to 0 to prevent them from being shifted around when many margin notes are used.

### B.4.3 Headers and Footers

The page style `plain` puts the page number into the footer in the right corner. When the option `twoside` is active, the page number of left pages is put into the left corner.

The pagestyles `headings` and `myheadings` create a header which spans the whole width of the page. The headings contain the running head (`\section` and `\subsection` in `refart` and `\chapter` and `\section` in `refrep`) when `headings` is used or a fixed text that can be defined with the `\markboth` command when `myheadings` is used. The heading will be set in a slanted font and separated from the body by a thin line.

In addition to the standard classes, `refman` supports a style for footers, which is used in this documentation. The information is exactly the same as in the headings but now printed in the footer with a thin line above.

To use a user-defined string you can say:

```
\pagestyle{myfootings} % or myheadings
\markboth{left title}{right title}
```

The `heading` and `myheading` commands increase the top margin by one line while the `footings` and `myfootings` commands decrease the top margin by one line. The page styles `empty` and `plain` leave the top margin unchanged. You should not combine headings and footings in one document.

User feedback has shown that it is not a good idea to combine `plain` and `(my)heading` either. Therefore I changed the layout of the `\chapter` page to `empty`. Maybe it is necessary to define a `hplain` and `fplain` pagestyle or to define some magic to use the correct definition of `plain`. Feedback is welcome.

## B.5 Additional commands

### B.5.1 Marginlabel

The command `\marginlabel{xxx}` prints the text `xxx` right justified into the left margin. Please note that a `\marginpar` will print it left justified.

Example: The word “Example” in the left margin is printed with the command `\marginlabel{Example:}`

### B.5.2 Attention

**! →** The command `\attention` puts an exclamation mark with an arrow pointing to the text into the left margin. This is an example for `\attention`.

**:-)** Since version 2.0c you can change the symbol used for the `\attention` command using a `\renewcommand{\attentionsymbol}{\texttt\{:-)\}}` command. To get the default back use `\renewcommand{\attentionsymbol}{\large \bfseries ! $\rightarrow$}`

**:-)** Since version 2.0c `\attention` takes an optional argument to define the symbol used in the margin. Thus you can change the symbol once, without having to restore it later. Do not abuse this feature, it is primarily

meant as an support for the `manfnt` package which enables you to use the “dangerous bend” and “double dangerous bend” signs.

The `manfnt` package is no longer enclosed with Refman, it has grown and is now a package of its own.

### B.5.3 Seealso

→ Chapter 1

The command `\seealso{n}` prints an arrow and its argument into the left margin. You will find examples for this in the left margin and in chapter 1.

### B.5.4 Maxipage environment

The `maxipage` environment is a special kind of `minipage` which extends over the full width of the page. It can be used for long formulas or `tabular` environments. You may use `maxipage` environments inside floats. You cannot use margin notes inside a `maxipage` and no page break will occur while in a `maxipage`. A `maxipage` is always a paragraph of its own with a thick line above and below. You can disable these lines with the `\maxipagerulefalse` command. They are on by default.

The following paragraph is an example for a `maxipage`:

---

This very long line is an example for a `maxipage`. It extends over the full width of the page, including the left margin.

---

This is normal text after the `maxipage`.

### B.5.5 Fullpage environment

The `fullpage` environment consists of one or more pages where the text extends over the full width of the page. You cannot use margin notes inside a `fullpage` environment. A `fullpage` will always start and end on a page of its own. It may be used for large tables, program listings or anything that does not fit into the normal layout.

→ Page 40

Page 40 is an example for a `fullpage`.

### B.5.6 Noparskip

The `\noparskip` removes the vertical space between two paragraphs. It is similar to the `\noindent` command that removes the indent of the first line of a paragraph.

### B.5.7 Setleftmarginwidth

The `\setleftmarginwidth` command is no longer supported. You can achieve similar results by using the `\setttextfraction` command.

### B.5.8 Descriptioncolon

By default a colon is printed after the description label. The command `\descriptioncolonfalse` disables the colon, the `\descriptioncolontrue` re-enables it.

### B.5.9 Descriptionleft

The `\descriptionlefttrue` command sets the description label left justified into the margin. The default is right justified which will be achieved with `\descriptionleftfalse`

### B.5.10 Maxipagerule

You can disable the rules before and after a `maxipage` with the `\maxipagerulefalse` command and re-enable them with the `\maxipageruletrue` command. The default is on. You should not mix `maxipages` with and without rules in one document.

### B.5.11 Condbreak

The command `\condbreak{2cm}` ensures, that the next 2 cm are either completely on this page or completely on the next. No page break will appear in the next 2 cm.

This is really a hack to achieve what the `\samepage` command often fails to do.

### B.5.12 Example

The `example` environment acts like a `verse` environment but uses a `tt` font.

### B.5.13 Pageperchapter

The command `\pageperchapter` creates page number that start with 1 for every new chapter. This may be useful for larger manuals. Since it works with chapters it is only available in the `refrep` class.

### B.5.14 Smallborder

The normal border around the page is 1 Inch. That is fine for a printed document, but wastes a lot of space when a document is meant for reading on screen. The option `smallborder` reduces the margin to 0.25 Inch.

You can redefine the border with `\setlength\papermarginwidth {0.25in}`. Call `\setpagefraction{0.7}` afterwards to recalculate the page layout.

### B.5.15 Dvips

The option `dvips` tells DVIPS about the current page size.

### B.5.16 Pdftex

The option `pdftex` tells PDF<sub>T</sub><sub>E</sub>X about the current page size.

### B.5.17 Pagesize

`pagesize` chooses the correct `\special`-command to tell the DVI-driver about the paper size. It works with DVIPS and DVIPDFMX for DVI output and PDF<sub>T</sub><sub>E</sub>X for PDF output.

### B.5.18 Ifpdfoutput

You can use `\ifpdfoutput{pdftext}{dvitext}` to write different text depending on the output format. This command was necessary to implement the `pagesize` option and is available for the user as well.

The last four commands have been taken from KOMA-Script, thanks Markus.

## Index

- attention, 43
- attentionsymbol, 43
- changing the layout, step by step, 33
- condbreak, 45
- description environment, 42
- descriptioncolon, 44
- descriptionleft, 45
- design, generic, 32
- design, logical, 32
- design, visual, 32
- designer, 9
- dvips, 45
- engineer, 9
- example, 45
- footings, 10, 38, 43
- footnote, 42
- fullpage, 44
- headings, 10, 38, 43
- ifpdfoutput, 46
- ipcnet design, 24
- ipcnet project, 24
- line length, 10
- line spacing, 10
- manfnt, 43
- margin notes, 10, 38
- marginlabel, 43
- maxipage, 44
- maxipagerule, 44, 45
- myfootings, 43
- myheadings, 43
- options, 41
- page design, horizontal, 41
- page design, vertical, 42
- page layout, 10
- pageperchapter, 45
- pagesize, 45
- pagestyle, 43
- papermarginwidth, 41
- pdftex, 45
- refart, invocation, 41
- refart.cls, 41
- refrep, invocation, 41
- refrep.cls, 41
- robowaifu design, 10
- rules, 10
- rules of thumb, documents, 10
- seealso, 44
- setleftmarginwidth, 44
- settfracfraction, 42
- smallborder, 45

Copyright (2020)

License (MIT) <https://opensource.org/licenses/MIT>