Contents:
AYABInterface Installation Instructions

1.1 Package installation from Pypi

The AYABInterface library requires Python 3. It can be installed from the Python Package Index.

1.1.1 Windows

Install it with a specific python version under windows:

```
py -3 -m pip --no-cache-dir install --upgrade AYABInterface
```

Test the installed version:

```
py -3 -m pytest --pyargs AYABInterface
```

1.1.2 Linux

To install the version from the python package index, you can use your terminal and execute this under Linux:

```
sudo python3 -m pip --no-cache-dir install --upgrade AYABInterface
```

Test the installed version:

```
python3 -m pytest --pyargs AYABInterface
```

1.2 Installation from Repository

You can setup the development version under Windows and Linux.

1.2.1 Linux

If you wish to get latest source version running, you can check out the repository and install it manually.
git clone https://github.com/fossasia/AYABInterface.git

cd AYABInterface

sudo python3 -m pip install --upgrade pip

sudo python3 -m pip install -r requirements.txt

sudo python3 -m pip install -r test-requirements.txt

py.test

To also make it importable for other libraries, you can link it into the site-packages folder this way:

sudo python3 setup.py link

### 1.2.2 Windows

Same as under *Linux* but you need to replace `sudo python3` with `py -3`. This also counts for the following documentation.
Development Setup

Make sure that you have the repository installed.

2.1 Install Requirements

To install all requirements for the development setup, execute

```
pip install --upgrade -r requirements.txt -r test-requirements.txt -r dev--_requirements.txt
```

2.2 Sphinx Documentation Setup

Sphinx was setup using the tutorial from readthedocs. It should be already setup if you completed the previous step.

Further reading:

- **domains**

With Notepad++ under Windows, you can run the `make_html.bat` file in the `docs` directory to create the documentation and show undocumented code.

2.3 Code Climate

To install the code climate command line interface (cli), read about it in their github repository. You need docker to be installed. Under Linux you can execute this in the Terminal to install docker:

```
wget -qO- https://get.docker.com/ | sh
sudo usermod -aG docker $USER
```

Then, log in and out. Then, you can install the command line interface:

```
wget -qO- https://github.com/codeclimate/codeclimate/archive/master.tar.gz | tar xvz
cd codeclimate-* && sudo make install
```

Then, go to the AYABInterface repository and analyze it.

```
codeclimate analyze
```
## 2.4 Version Pinning

We use version pinning, described in this blog post (outdated). Also read the current version for how to set up.

After installation you can run:

```bash
pip install -r requirements.in -r test-requirements.in -r dev-requirements.in
pip-compile --output-file requirements.txt requirements.in
pip-compile --output-file test-requirements.txt test-requirements.in
pip-compile --output-file dev-requirements.txt dev-requirements.in
pip-sync requirements.txt dev-requirements.txt test-requirements.txt
pip install --upgrade -r requirements.txt -r test-requirements.txt -r dev-
˓--requirements.txt

pip-sync uninstalls every package you do not need and writes the fix package versions to the requirements files.
```

## 2.5 Continuous Integration to Pypi

Before you put something on Pypi, ensure the following:

1. The version is in the master branch on github.
2. The tests run by travis-ci run successfully.

Pypi is automatically deployed by travis. See here. To upload new versions, tag them with git and push them.

```bash
setup.py tag_and_deploy
```

The tag shows up as a travis build. If the build succeeds, it is automatically deployed to Pypi.

## 2.6 Manual Upload to the Python Package Index

However, here you can see how to upload this package manually.

### 2.6.1 Version

Throughout this chapter, `<new_version>` refers to a string of the form `[0-9]+\.[0-9]+\.[0-9]+[ab]` or `<MAYOR>.<MINOR>.<STEP>[<MATURITY>]` where `<MAYOR>`, `<MINOR>` and `<STEP>` represent numbers and `<MATURITY>` can be a letter to indicate how mature the release is.

1. Create a new branch for the version.

```bash
git checkout -b <new_version>
```

2. Increase the `__version__` in `__init__.py`
   - no letter at the end means release
   - b in the end means Beta
   - a in the end means Alpha

3. Commit and upload this version.
4. Create a pull-request.
5. Wait for travis-ci to pass the tests.
6. Merge the pull-request.
7. Checkout the master branch and pull the changes from the commit.

```bash
git checkout master
git pull
```
8. Tag the version at the master branch with a v in the beginning and push it to github.

```bash
git tag v<new_version>
git push origin v<new_version>
```
9. Upload the code to Pypi.

### 2.6.2 Upload

First ensure all tests are running:

```bash
setup.py pep8
```

From docs.python.org:

```bash
setup.py sdist bdist_wininst upload register
```

### 2.7 Classifiers

You can find all Pypi classifiers [here](#).
3.1 The AYABInterface Module Reference

3.1.1 AYABInterface Module

AYABInterface - a module to control the AYAB shield.

See also:
http://ayab-knitting.com/

AYABInterface. NeedlePositions ( *args, **kw)
Create a new NeedlePositions object.

Returns an AYABInterface.needle_positions.NeedlePositions

See also:
AYABInterface.needle_positions.NeedlePositions

AYABInterface. get_machines ()
Return a list of all machines that can be used.

Return type list

Returns a list of Machines

AYABInterface. get_connections ()
Return a list of all available serial connections.

Return type list

Returns a list of AYABInterface.SerialPort. All of the returned objects have a
connect() method and a name attribute.

3.1.2 actions Module

These are the actions that can be executed by the users.

class AYABInterface.actions. ActionMetaClass ( name, bases, attributes)
Bases: type

Metaclass for the actions.

This class makes sure each Action has tests.
If a class is named `MyAction`, each `Action` gets the method `is_my_action()` which returns `False` for all `Actions` except for `MyAction` it returns `True`.

```python
__init__( name, bases, attributes)
Create a new `Action` subclass.
```
class AYABInterface.actions.Action (*arguments)
Bases: object
A base class for actions.

```python
__eq__( other)
Whether this object is equal to the other.

Return type bool
```
```python
__hash__()  
The hash of the object.

Return type int

Returns the `hash()` of the object
```

```python
__init__( *arguments)
Create a new `Action`.

Parameters arguments (tuple) – The arguments passed to the action. These are also used to determine `equality` and the `hash`.
```
```python
__repr__()
Return this object as string.

Return type str
```
```python
__weakref__
list of weak references to the object (if defined)
```
```python
is_action ()
Test whether this is a Action.

Return type bool

Returns True
```
```python
is_move_carriage_over_left_hall_sensor ()
Test whether this is a MoveCarriageOverLeftHallSensor.

Return type bool

Returns False
```
```python
is_move_carriage_to_the_left ()
Test whether this is a MoveCarriageToTheLeft.

Return type bool

Returns False
```
```python
is_move_carriage_to_the_right ()
Test whether this is a MoveCarriageToTheRight.

Return type bool

Returns False
```
```python
is_move_needles_into_position ()
Test whether this is a MoveNeedlesIntoPosition.
```
Return type  bool
Returns  False

is_put_color_in_nut_a()  
Test whether this is a PutColorInNutA.

Return type  bool
Returns  False

is_put_color_in_nut_b()  
Test whether this is a PutColorInNutB.

Return type  bool
Returns  False

is_switch_carriage_to_mode_kc()  
Test whether this is a SwitchCarriageToModeKc.

Return type  bool
Returns  False

is_switch_carriage_to_mode_nl()  
Test whether this is a SwitchCarriageToModeNl.

Return type  bool
Returns  False

is_switch_off_machine()  
Test whether this is a SwitchOffMachine.

Return type  bool
Returns  False

is_switch_on_machine()  
Test whether this is a SwitchOnMachine.

Return type  bool
Returns  False

class  AYABInterface.actions.SwitchCarriageToModeKc(*arguments)
Bases:  AYABInterface.actions.Action

The user switches the mode of the carriage to KC.

is_switch_carriage_to_mode_kc()  
Test whether this is a SwitchCarriageToModeKc.

Return type  bool
Returns  True

class  AYABInterface.actions.SwitchCarriageToModeNl(*arguments)
Bases:  AYABInterface.actions.Action

The user switches the mode of the carriage to NL.

is_switch_carriage_to_mode_nl()  
Test whether this is a SwitchCarriageToModeNl.

Return type  bool
class AYABInterface.actions.MoveCarriageOverLeftHallSensor (*arguments)
    Bases: AYABInterface.actions.Action

    The user moves the carriage over the left hall sensor.

    is_move_carriage_over_left_hall_sensor ()
        Test whether this is a MoveCarriageOverLeftHallSensor.

    Return type: bool

    Returns: True

class AYABInterface.actions.MoveCarriageToTheLeft (*arguments)
    Bases: AYABInterface.actions.Action

    The user moves the carriage to the left.

    is_move_carriage_to_the_left ()
        Test whether this is a MoveCarriageToTheLeft.

    Return type: bool

    Returns: True

class AYABInterface.actions.MoveCarriageToTheRight (*arguments)
    Bases: AYABInterface.actions.Action

    The user moves the carriage to the right.

    is_move_carriage_to_the_right ()
        Test whether this is a MoveCarriageToTheRight.

    Return type: bool

    Returns: True

class AYABInterface.actions.PutColorInNutA (*arguments)
    Bases: AYABInterface.actions.Action

    The user puts a color into nut A.

    is_put_color_in_nut_a ()
        Test whether this is a PutColorInNutA.

    Return type: bool

    Returns: True

class AYABInterface.actions.PutColorInNutB (*arguments)
    Bases: AYABInterface.actions.Action

    The user puts a color into nut B.

    is_put_color_in_nut_b ()
        Test whether this is a PutColorInNutB.

    Return type: bool

    Returns: True

class AYABInterface.actions.MoveNeedlesIntoPosition (*arguments)
    Bases: AYABInterface.actions.Action

    The user moves needles into position.
```python
is_move_needs_into_position()  
Test whether this is a MoveNeedlesIntoPosition.  
    Return type bool  
    Returns True

class AYABInterface.actions.SwitchOffMachine(*arguments)  
Bases: AYABInterface.actions.Action  
The user switches off the machine.  
    is_switch_off_machine()  
    Test whether this is a SwitchOffMachine.  
    Return type bool  
    Returns True

class AYABInterface.actions.SwitchOnMachine(*arguments)  
Bases: AYABInterface.actions.Action  
The user switches on the machine.  
    is_switch_on_machine()  
    Test whether this is a SwitchOnMachine.  
    Return type bool  
    Returns True

3.1.3 carriages Module

This module contains all the supported carriages.

class AYABInterface.carriages.Carriage  
Bases: object  
A base class for carriages.  
    __eq__(other)  
    Equivalent to self == other.  
    __hash__()  
    Make this object hashable.  
    __repr__()  
    This object as string.  
    __weakref__  
    list of weak references to the object (if defined)

class AYABInterface.carriages.KnitCarriage  
Bases: AYABInterface.carriages.Carriage  
The carriage used for knitting.

3.1.4 interaction Module

This module can be used to interact with the AYAB Interface.
```
class AYABInterface.interaction. Interaction (knitting_pattern, machine)
    Bases: object

Interaction with the knitting pattern.

__init__ (knitting_pattern, machine)
    Create a new interaction object.

Parameters

• knitting_pattern – a KnittingPattern
• machine (AYABInterface.machines.Machine) – the machine to knit on

__weakref__
    list of weak references to the object (if defined)

actions
    A list of actions to perform.

    Returns a list of AYABInterface.actions.Action

communicate_through (file)
    Setup communication through a file.

    Return type AYABInterface.communication.Communication

communication
    The communication with the controller.

    :rtype: AYABInterface.communication.Communication

3.1.5 machines Module

This module contains the information about the different types of machines.

Every machine specific knowledge should be put in this file. Machine specific knowledge is, for example:

• the number of needles a machine supports
• whether it is single or double bed
• how many colors are supported
• the name of the machine

class AYABInterface.machines. Machine
    Bases: object

The type of the machine.

This is an abstract base class and some methods need to be overwritten.

NAME = None
    the name of the machine

__eq__ (other)
    Equivalent of self == other.

    Return type bool

    Returns whether this object is equal to the other object
__hash__ ( )
    Return the hash of this object.

See also:
    hash()

__repr__ ( )
    Return this object as a string.

__weakref__
    list of weak references to the object (if defined)

is_ck35 ( )
    Whether this machine is a Brother CK-35.
    
    Return type  bool

is_kh270 ( )
    Whether this machine is a Brother KH-270.
    
    Return type  bool

is_kh900 ( )
    Whether this machine is a Brother KH-910.
    
    Return type  bool

is_kh910 ( )
    Whether this machine is a Brother KH-900.
    
    Return type  bool

is_kh930 ( )
    Whether this machine is a Brother KH-930.
    
    Return type  bool

is_kh950 ( )
    Whether this machine is a Brother KH-950.
    
    Return type  bool

is_kh965 ( )
    Whether this machine is a Brother KH-965.
    
    Return type  bool

left_end_needle
    The index of the leftmost needle.
    
    Return type  int
    
    Returns  0

name
    The identifier of the machine.

needle_positions
    The different needle positions.
    
    Return type  tuple

needle_positions_to_bytes ( needle_positions )
    Convert the needle positions to the wire format.
    
    This conversion is used for The cnfLine Message.
Parameters

**needle_positions** – an iterable over `needle_positions` of length `number_of_needles`

*Return type* bytes

**number_of_needles**

The number of needles of this machine.

*Return type* int

**right_end_needle**

The index of the rightmost needle.

*Return type* int

*Returns* `left_end_needle + number_of_needles - 1`

### class AYABInterface.machines.KH9XXSeries

Bases: AYABInterface.machines.Machine

The base class for the KH9XX series.

**needle_positions**

The different needle positions.

*Return type* tuple

*Returns* the needle positions are “B” and “D”

**number_of_needles**

The number of needles on this machine.

*Return type* int

*Returns* 200. The KH9XX series has 200 needles.

### class AYABInterface.machines.CK35

Bases: AYABInterface.machines.Machine

The machine type for the Brother CK-35.

**needle_positions**

The different needle positions.

*Return type* tuple

*Returns* the needle positions are “B” and “D”

**number_of_needles**

The number of needles on this machine.

*Return type* int

*Returns* 200. The KH9XX series has 200 needles.

### class AYABInterface.machines.KH900

Bases: AYABInterface.machines.KH9XXSeries

The machine type for the Brother KH-900.

### class AYABInterface.machines.KH910

Bases: AYABInterface.machines.KH9XXSeries

The machine type for the Brother KH-910.
class AYABInterface.machines.KH930
    Bases: AYABInterface.machines.KH9XXSeries
    The machine type for the Brother KH-930.

class AYABInterface.machines.KH950
    Bases: AYABInterface.machines.KH9XXSeries
    The machine type for the Brother KH-950.

class AYABInterface.machines.KH965
    Bases: AYABInterface.machines.KH9XXSeries
    The machine type for the Brother KH-965.

class AYABInterface.machines.KH270
    Bases: AYABInterface.machines.Machine
    The machine type for the Brother KH-270.

    needle_positions
        The different needle positions.
        Return type tuple
        Returns the needle positions are “B” and “D”

    number_of_needles
        The number of needles on this machine.
        Return type int
        Returns 200. The KH9XX series has 200 needles.

AYABInterface.machines.get_machines()
    Return a list of all machines.
    Return type list
    Returns a list of Machines

3.1.6 needle_positions Module

This module provides the interface to the AYAB shield.

class AYABInterface.needle_positions.NeedlePositions ( rows, machine)
    Bases: object
    An interface that just focusses on the needle positions.

    __init__ ( rows, machine)
        Create a needle interface.

        Parameters
            • rows (list) – a list of lists of needle positions
            • Machine – the machine type to use

        Raises ValueError – if the arguments are not valid, see check ()

    __weakref__
        list of weak references to the object (if defined)

    check ()
        Check for validity.
Raises `ValueError` –
  • if not all lines are as long as the number of needles
  • if the contents of the rows are not needle positions

`completed_row_indices`
  The indices of the completed rows.
  
  **Return type** list
  When a row was completed, the index of the row turns up here. The order is preserved, entries may occur duplicated.

`get_row` *(index, default=None)*
  Return the row at the given index or the default value.

`machine`
  The machine these positions are on.

`on_row_completed` *(callable)*
  Add an observer for completed rows.
  
  **Parameters** callable – a callable that is called with the row index as first argument
  When row_completed() was called, this callable is called with the row index as first argument. Call this method several times to register more observers.

`row_completed` *(index)*
  Mark the row at index as completed.

  **See also:**
  `completed_row_indices()`
  This method notifies the observers from on_row_completed().

### 3.1.7 serial Module

The serial interface.

Execute this module to print all serial ports currently available.

AYABInterface.serial. `list_serial_port_strings()`
  Lists serial port names.
  
  **Raises** EnvironmentError – On unsupported or unknown platforms
  **Returns** A list of the serial ports available on the system
  
  **See also:**
  The Stack Overflow answer

AYABInterface.serial. `list_serial_ports()`
  Return a list of all available serial ports.
  
  **Return type** list
  **Returns** a list of serial ports

class AYABInterface.serial. `SerialPort` *(port)*
  Bases: object
  A class abstracting the port behavior.
__init__(port)
Create a new serial port instance.

Parameters port (str) – the port to connect to

Note: The baud rate is specified in Serial Communication

__repr__()
Return this object as string.

Return type str

__weakref__
list of weak references to the object (if defined)

connect()
Return a connection to this port.

Return type serial.Serial

name
The name of the port for displaying.

Return type str

3.1.8 utils Module

Utility methods.

AYABInterface.utils.sum_all(iterable, start)
Sum up an iterable starting with a start value.

In contrast to sum(), this also works on other types like lists and sets.

AYABInterface.utils.number_of_colors(rows)
Determine the number of colors in the rows.

Return type int

AYABInterface.utils.next_line(last_line, next_line_8bit)
Compute the next line based on the last line and a 8bit next line.

The behaviour of the function is specified in The reqLine Message.

Parameters

• last_line(int) – the last line that was processed

• next_line_8bit(int) – the lower 8 bits of the next line

Returns the next line closest to last_line

See also:
The reqLine Message

AYABInterface.utils.camel_case_to_under_score(camel_case_name)
Return the underscore name of a camel case name.

Parameters camel_case_name(str) – a name in camel case such as "ACamelCaseName"

Returns the name using underscores, e.g. "a_camel_case_name"

Return type str
3.2 The AYABInterface.convert Module Reference

3.2.1 convert Module

Conversion of colors to needle positions.

AYABInterface.convert.colors_to_needle_positions(rows)

Convert rows to needle positions.

Returns
Return type  list

class AYABInterface.convert.NeedlePositions(needle_coloring, colors, two_colors)
Bases: tuple

__getnewargs__()  
Return self as a plain tuple. Used by copy and pickle.

__getstate__()  
Exclude the OrderedDict from pickling

static __new__(_cls, needle_coloring, colors, two_colors)
Create new instance of NeedlePositions(needle_coloring, colors, two_colors)

__repr__()  
Return a nicely formatted representation string

colors  
Alias for field number 1

needle_coloring  
Alias for field number 0

two_colors  
Alias for field number 2

3.3 The AYABInterface.communication Module Reference

3.3.1 communication Module

This module is used to communicate with the shield.

Requirement: Make objects from binary stuff.

class AYABInterface.communication.Communication(file, get_needle_positions, machine, on_message_received=(), left_end_needle=None, right_end_needle=None)
Bases: object

This class communicates with the AYAB shield.

__init__(file, get_needle_positions, machine, on_message_received=(), left_end_needle=None, right_end_needle=None)
Create a new Communication object.

Parameters
• **file** – a file-like object with read and write methods for the communication with the Arduino. This could be a `serial.Serial` or a `socket.socket.makefile()`.

• **get_needle_positions** – a callable that takes an `index` and returns `None` or an iterable over needle positions.

• **machine** (`AYABInterface.machines.Machine`) – the machine to use for knitting

• **on_message_received** (`list`) – an iterable over callables that takes a received `message` whenever it comes in. Since `state` changes only take place when a message is received, this can be used as an state observer.

• **left_end_needle** – A needle number on the machine. Other needles that are on the left side of this needle are not used for knitting. Their needle positions are not be set.

• **right_end_needle** – A needle number on the machine. Other needles that are on the right side of this needle are not used for knitting. Their needle positions are not be set.

---

**__weakref__**

- list of weak references to the object (if defined)

**api_version_is_supported** `(api_version)`

Return whether an api version is supported by this class.

Return type  `bool`

Returns if the `api_version` is supported

Parameters  `api_version` (`int`) – the api version

Currently supported api versions: 4

**can_receive_messages** `()`

Whether this communication is ready to receive messages.

Return type  `bool`

```
assert not communication.can_receive_messages()
communication.start()
assert communication.can_receive_messages()
communication.stop()
assert not communication.can_receive_messages()
```

**controller**

Information about the controller.

If no information about the controller is received, the return value is `None`.

If information about the controller is known after *The cnfInfo Message* was received, you can access these values:

```
>>> communication.controller.firmware_version
(5, 2)
>>> communication.controller.firmware_version.major
5
>>> communication.controller.firmware_version.minor
2
>>> communication.controller.api_version
4
```

**last_requested_line_number**

The number of the last line that was requested.
left_end_needle
The left end needle of the needle positions.

Return type int
Returns the left end needle of the machine

lock
The lock of the communication.

In case you parallelize() the communication, you may want to use this lock to make sure the parallelization does not break your code.

needle_positions
A cache for the needle positions.

Return type AYABInterface.communication.cache.NeedlePositionCache

on_message (callable)
Add an observer to received messages.

Parameters callable – a callable that is called every time a
AYABInterface.communication.host_messages.Message is sent or a
AYABInterface.communication.controller_messages.Message is received

parallelize (seconds_to_wait=2)
Start a parallel thread for receiving messages.

If start() was no called before, start will be called in the thread. The thread calls receive_message() until the state is_connection_closed().

Parameters seconds_to_wait (float) – A time in seconds to wait with the parallel execution. This is useful to allow the controller time to initialize.

See also:
lock, runs_in_parallel()

receive_message ()
Receive a message from the file.

right_end_needle
The left end needle of the needle positions.

Return type int
Returns the right end needle of the machine

runs_in_parallel ()
Whether the communication runs in parallel.

Return type bool
Returns whether parallelize() was called and the communication still receives messages and is not stopped

send (host_message_class, *args)
Send a host message.

Parameters
• host_message_class (type) – a subclass of AYABInterface.communication.host_messages.Message

• args – additional arguments that shall be passed to the host_message_class as arguments

start ()
Start the communication about a content.

Parameters content (Content) – the content of the communication.

state
The state this object is in.

Returns the state this communication object is in.

Return type AYABInterface.communication.states.State

Note: When calling parallelize() the state can change while you check it.

stop ()
Stop the communication with the shield.

3.3.2 cache Module

Convert and cache needle positions.

class AYABInterface.communication.cache.NeedlePositionCache (get_needle_positions, machine)

Bases: object

Convert and cache needle positions.

__init__ (get_needle_positions, machine)
Create a new NeedlePositions object.

__weakref__
list of weak references to the object (if defined)

get (line_number)
Return the needle positions or None.

Parameters line_number (int) – the number of the line

Return type list

Returns the needle positions for a specific line specified by line_number or None if no were given

get_bytes (line_number)
Get the bytes representing needle positions or None.

Parameters line_number (int) – the line number to take the bytes from

Return type bytes

Returns the bytes that represent the message or None if no data is there for the line.

Depending on the machine, the length and result may vary.

g_get_line_configuration_message (line_number)
Return the cnfLine content without id for the line.
Parameters `line_number` (`int`) – the number of the line

Return type `bytes`

Returns a cnfLine message without id as defined in *The cnfLine Message*

`is_last` (`line_number`)  
Whether the line number is has no further lines.

Return type `bool`

Returns is the next line above the line number are not specified

### 3.3.3 Carrieages Module

This module contains the carriages which are communicated by the firmware.

```python
class AYABInterface.communication.carriages.NullCarriage (needle_position)  
Bases: AYABInterface.communication.carriages.Carriage
This is an empty carriage.
class AYABInterface.communication.carriages.KnitCarriage (needle_position)  
Bases: AYABInterface.communication.carriages.Carriage
The carriage for knitting.

is_knit_carriage ()  
This is a knit carriage.

Return type `bool`

Returns True

class AYABInterface.communication.carriages.HoleCarriage (needle_position)  
Bases: AYABInterface.communication.carriages.Carriage
The carriage for creating holes.

is_hole_carriage ()  
This is a knit carriage.

Return type `bool`

Returns True

class AYABInterface.communication.carriages.UnknownCarriage (needle_position)  
Bases: AYABInterface.communication.carriages.Carriage
The carriage type if the type is not known.

is_unknown_carriage ()  
The type of this carriage is unknown.

Return type `bool`

Returns True

AYABInterface.communication.carriages.id_to_carriage_type (carriage_id)  
Returns the carriage type for an id.

Return type `type`

Returns a subclass of `Carriage`
```
class AYABInterface.communication.carriages.Carriage(needle_position)
    Bases: object
    A base class for carriages.
    __init__(needle_position)
        Create a new carriage.
        Parameters needle_position(int) – the position of the carriage
    __weakref__
        list of weak references to the object (if defined)
    is_hole_carriage()
        Whether this is a hole carriage.
        Return type bool
        Returns False
    is_knit_carriage()
        Whether this is a knit carriage.
        Return type bool
        Returns False
    is_unknown_carriage()
        Whether the type of this carriage is unknown.
        Return type bool
        Returns False
    needle_position
        The needle position of the carriages.
        Returns the needle position of the carriage counted from the left, starting with 0
        Return type int

3.3.4 hardware_messages Module

This module contains all the messages that are received.
AYABInterface.communication.hardware_messages.read_message_type(file)
    Read the message type from a file.

class AYABInterface.communication.hardware_messages.StateIndication(file, communication)
    Bases: AYABInterface.communication.hardware_messages.FixeSizeMessage
    This message shows the state of the controller.

See also:
The indState Message
MESSAGE_ID = 132
    The first byte that indicates this message
    carriage
        The carriage which is reported.
Return type: `AYABInterface.communication.carriages.Carriage`

Returns: the carriage with information about its position

**current_needle**
The current needle position.

Return type: `int`

**is_ready_to_knit()**
Whether this message indicates that the controller can knit now.

**is_state_indication()**
Whether this is a InformationConfirmation message.

Return type: `bool`

Returns: True

is_valid()
Whether this messages matches the specification.

**left_hall_sensor_value**
The value of the left hall sensor.

Return type: `int`

**received_by(visitor)**
Call visitor.receive_state_indication.

**right_hall_sensor_value**
The value of the left hall sensor.

Return type: `int`

---

class AYABInterface.communication.hardware_messages.LineRequest(file, communication)

Bases: `AYABInterface.communication.hardware_messages.FixedSizeMessage`

The controller requests a line.

See also:
*The reqLine Message*

**MESSAGE_ID = 130**
The first byte that indicates this message

**is_line_request()**
Whether this is a LineRequest message.

Return type: `bool`

Returns: True

**line_number**
The line number that was requested.

**received_by(visitor)**
Call visitor.receive_line_request.

---

class AYABInterface.communication.hardware_messages.TestConfirmation(file, communication)

Bases: `AYABInterface.communication.hardware_messages.SuccessConfirmation`
This message is sent at/when

**MESSAGE_ID = 196**
The first byte that indicates this message

**is_test_confirmation ()**
Whether this is a TestConfirmation message.

Return type **bool**

Returns **True**

**received_by (visitor)**
Call visitor.test_information_confirmation.

```python
class AYABInterface.communication.hardware_messages. InformationConfirmation (file, communication)
```

**Bases:** `AYABInterface.communication.hardware_messages.FixedSizeMessage`

This message is the answer in the initial handshake.

A `InformationRequest` requests this message from the controller to start the initial handshake.

See also:

The `cnfInfo Message InformationRequest`

**MESSAGE_ID = 195**
The first byte that indicates this message

**api_version**
The API version of the controller.

Return type **int**

**api_version_is_supported ()**
Whether the communication object supports this API version.

Return type **bool**

See also:

`Communication.api_version_is_supported`

**firmware_version**
The firmware version of the controller.

Return type **FirmwareVersion**

```python
minor_version = int()
mayor_version = int()
assert message.firmware_version == (mayor_version, minor_version)
assert message.firmware_version.major == mayor_version
assert message.firmware_version.minor == minor_version
```

**is_information_confirmation ()**
Whether this is a InformationConfirmation message.

Return type **bool**
Reach True

received_by (visitor)
Call visitor.receive_information_confirmation.

class AYABInterface.communication.hardware_messages. Debug (file, communication)
Bases: AYABInterface.communication.hardware_messages.Message

This message contains debug output from the controller.

See also:
The debug Message

bytes
The debug message as bytes.

Return type bytes

Returns the debug message as bytes without the b'
' at the end

is_debug ()
Whether this is a Debug message.

Return type bool

Returns False

received_by (visitor)
Call visitor.receive_debug.

class AYABInterface.communication.hardware_messages. StartConfirmation (file, communication)
Bases: AYABInterface.communication.hardware_messages.SuccessConfirmation

This marks the success or failure of a reqStart message.

See also:
The cnfStart Message

MESSAGE_ID = 193
The first byte that indicates this message

is_start_confirmation ()
Whether this is a StartConfirmation message.

Return type bool

Returns True

received_by (visitor)
Call visitor.receive_state_confirmation.

class AYABInterface.communication.hardware_messages. SuccessConfirmation (file, communication)
Bases: AYABInterface.communication.hardware_messages.FixedSizeMessage

Base class for massages of success and failure.
is_success ()
    Whether the configuration was successful.
    
    Return type  bool

is_valid ()
    Whether this message is valid.

class AYABInterface.communication.hardware_messages.UnknownMessage (file, communication)

Bases: AYABInterface.communication.hardware_messages.FixedSizeMessage

This is a special message for unknown message types.

is_unknown ()
    Whether this is a StateIndication message.
    
    Return type  bool
    
    Returns  True

is_valid ()
    Whether the message is valid.
    
    Return type  bool
    
    Returns  False

received_by (visitor)
    Call visitor.receive_unknown.

class AYABInterface.communication.hardware_messages.Message (file, communication)

Bases: object

This is the base class for messages that are received.

__init__ (file, communication)
    Create a new Message.

__repr__ ()
    This object as string.
    
    Return type  str

__weakref__
    list of weak references to the object (if defined)

is_connection_closed ()
    Whether this is a ConnectionClosed message.
    
    Return type  bool
    
    Returns  False

is_debug ()
    Whether this is a Debug message.
    
    Return type  bool
    
    Returns  False

is_from_controller ()
    Whether this message is sent by the controller.
    
    Return type  bool
Returns True

is_from_host ()
  Whether this message is sent by the host.
  Return type bool
  Returns False

is_information_confirmation ()
  Whether this is a InformationConfirmation message.
  Return type bool
  Returns False

is_line_request ()
  Whether this is a LineRequest message.
  Return type bool
  Returns False

is_start_confirmation ()
  Whether this is a StartConfirmation message.
  Return type bool
  Returns False

is_state_indication ()
  Whether this is a StateIndication message.
  Return type bool
  Returns False

is_test_confirmation ()
  Whether this is a TestConfirmation message.
  Return type bool
  Returns False

is_unknown ()
  Whether this is a StateIndication message.
  Return type bool
  Returns False

is_valid ()
  Whether the message is valid.
  Return type bool
  Returns True

wants_to_answer ()
  Whether this message produces and answer message.
  Return type bool
  Returns False
class AYABInterface.communication.hardware_messages.ConnectionClosed (file, 
communication)

Bases: AYABInterface.communication.hardware_messages.Message

This message is notified about when the connection is closed.

is_connection_closed ()

Whether this is a ConnectionClosed message.

Return type bool

Returns True

received_by (visitor)

Call visitor.receive_connection_closed.

class AYABInterface.communication.hardware_messages.FirmwareVersion (major, minor)

Bases: tuple

__getnewargs__ ()

Return self as a plain tuple. Used by copy and pickle.

__getstate__ ()

Exclude the OrderedDict from pickling

static __new__ (cls, major, minor)

Create new instance of FirmwareVersion(major, minor)

__repr__ ()

Return a nicely formatted representation string

major

Alias for field number 0

minor

Alias for field number 1

class AYABInterface.communication.hardware_messages.FixedSizeMessage (file, 
communication)

Bases: AYABInterface.communication.hardware_messages.Message

This is a message of fixed size.

__init__ (file, communication)

Create a new Message.

read_end_of_message ()

Read the b"rn" at the end of the message.

3.3.5 host_messages Module

This module contains the messages that are sent to the controller.

class AYABInterface.communication.host_messages.Message (file, communication, *args, 
**kw)

Bases: object

This is the interface for sent messages.
MESSAGE_ID = None
    the first byte to identify this message
__init__ (file, communication, *args, **kw)
    Create a new Request object
__repr__ ()
    This message as string including the bytes.
    Return type: str
__weakref__
    list of weak references to the object (if defined)
as_bytes ()
    The message represented as bytes.
    Return type: bytes
content_bytes ()
    The message content as bytes.
    Return type: bytes
init ()
    Override this method.
is_from_controller ()
    Whether this message is sent by the controller.
    Return type: bool
    Returns: False
is_from_host ()
    Whether this message is sent by the host.
    Return type: bool
    Returns: True
send ()
    Send this message to the controller.
class AYABInterface.communication.host_messages.StartRequest (file, communication,
            *args, **kw)
    Bases: AYABInterface.communication.host_messages.Message
    This is the start of the conversation.
See also:
The reqStart Message
MESSAGE_ID = 1
    the first byte to identify this message
content_bytes ()
    Return the start and stop needle.
    Return type: bytes
init (left_end_needle, right_end_needle)
    Initialize the StartRequest with start and stop needle.
• **TypeError** – if the arguments are not integers

• **ValueError** – if the values do not match the *specification*

**left_end_needle**  
The needle to start knitting with.

  
  **Return type** int  
  **Returns** value where $0 \leq value \leq 198$

**right_end_needle**  
The needle to start knitting with.

  
  **Return type** int  
  **Returns** value where $1 \leq value \leq 199$

**class** `AYABInterface.communication.host_messages.LineConfirmation(file, communication, *args,**kw)`

  **Bases:** `AYABInterface.communication.host_messages.Message`

This message send the data to configure a line.

**See also:**

*The cnfLine Message*

**MESSAGE_ID** = 66  
the first byte to identify this message

**content_bytes()**  
Return the start and stop needle.

**init(line_number)**  
Initialize the StartRequest with the line number.

**class** `AYABInterface.communication.host_messages.InformationRequest(file, communication, *args,**kw)`

  **Bases:** `AYABInterface.communication.host_messages.Message`

Start the initial handshake.

**See also:**

*The reqInfo Message, InformationConfirmation*

**MESSAGE_ID** = 3  
the first byte to identify this message

**class** `AYABInterface.communication.host_messages.TestRequest(file, communication, *args,**kw)`

  **Bases:** `AYABInterface.communication.host_messages.Message`

Start the test mode of the controller.

**See also:**

*The reqTest Message, InformationConfirmation*

**MESSAGE_ID** = 4  
the first byte to identify this message
3.3.6 states Module

This module contains the state machine for the communication class.

Click on this image to go to the states from the diagram:

```python
class AYABInterface.communication.states.State(communication)
    Bases: object

    The base class for states.

    __init__(communication)
    Create a new state.

    Please use the subclasses of this.

    Parameters communication (AYABInterface.communication.Communication)
        – the communication object which is in this state

    __repr__()
    This object as string.

    Return type: str

    __weakref__
    list of weak references to the object (if defined)

    communication_started()
    Call when the communication starts.

    enter()
    Called when the state is entered.

    The AYABInterface.communication.Communication.state is set to this state.

    exit()
    Called when this state is left.

    The AYABInterface.communication.Communication.state is set to this state.

    is_before_knitting()
    Whether the knitting should start soon.

    Return type: bool

    Returns: False

    is_connection_closed()
    Whether the connection is closed.

    Return type: bool

    Returns: False

    is_final()
    Whether the communication is over.

    Return type: bool

    Returns: False

    is_initial_handshake()
    Whether the communication object is in the initial handshake.

    Return type: bool

    Returns: False
```
is_initializing_machine()  
Whether the machine is currently being initialized.
  
  Return type: bool  
  Returns: False

is_knitting()  
Whether the machine ready to knit or knitting.
  
  Return type: bool  
  Returns: False

is_knitting_line()  
Whether the machine knits a line.
  
  Return type: bool  
  Returns: False

is_knitting_started()  
Whether the machine ready to knit the first line.
  
  Return type: bool  
  Returns: false

is_starting_to_knit()  
Whether the machine initialized and knitting starts.
  
  Return type: bool  
  Returns: False

is_unsupported_api_version()  
Whether the API version of communication and controller do not match.
  
  Return type: bool  
  Returns: False

is_waiting_for_start()  
Whether this state is waiting for the start.
  
  Return type: bool  
  Returns: False

is_waiting_for_the_communication_to_start()  
Whether the communication can be started.
  
  When this is True, you call `AYABInterface.communication.Communication.start()` to leave the state.
  
  Return type: bool  
  Returns: False

receive_connection_closed(message)  
Receive a ConnectionClosed message.
  
  Parameters: message - a `ConnectionClosed` message  
  If the is called, the communication object transits into the `ConnectionClosed`.

receive_debug(message)  
Receive a Debug message.
Parameters `message` – a `Debug` message

This logs the debug message.

`receive_information_confirmation (message)`
Receive a `InformationConfirmation` message.

Parameters `message` – a `InformationConfirmation` message

`receive_line_request (message)`
Receive a `LineRequest` message.

Parameters `message` – a `LineRequest` message

`receive_message (message)`
Receive a message from the controller.

Parameters `message (AYABInterface.communication.hardware_messages.Message)` – the message to receive

This method calls `message.received_by` which dispatches the call to the `receive_*` methods.

`receive_start_confirmation (message)`
Receive a `StartConfirmation` message.

Parameters `message` – a `StartConfirmation` message

`receive_state_indication (message)`
Receive a `StateIndication` message.

Parameters `message` – a `StateIndication` message

`receive_test_confirmation (message)`
Receive a `TestConfirmation` message.

Parameters `message` – a `TestConfirmation` message

`receive_unknown (message)`
Receive a `UnknownMessage` message.

Parameters `message` – a `UnknownMessage` message

The connection is closed.

`is_connection_closed ()`
The connection is closed.

Return type `bool`

Returns `True`

The connection is closed.

This is the initial state of a `AYABInterface.communication.Communication`.

`communication_started ()`
Call when the communication starts.

The communication object transits into `InitialHandshake`.
is_before_knitting ()
Whether the knitting should start soon.

Return type bool
Returns True

is_waiting_for_start ()
Whether this state is waiting for the start.

Call AYABInterface.communication.Comunication.start() to leave this state.

Return type bool
Returns True

class AYABInterface.communication.states. InitialHandshake (communication)
Bases: AYABInterface.communication.states.State

The communication has started.

enter ()
This starts the handshake.

A AYABInterface.communication.host_messages.InformationRequest is sent to the controller.

is_before_knitting ()
Whether the knitting should start soon.

Return type bool
Returns True

is_initial_handshake ()
Whether the communication object is in the intial handshake.

Return type bool
Returns True

receive_information_confirmation (message)
A InformationConfirmation is received.

If the api version is supported, the communication object transitions into a InitializingMachine, if unsupported, into a UnsupportedApiVersion

class AYABInterface.communication.states. UnsupportedApiVersion (communication)
Bases: AYABInterface.communication.states.FinalState

The api version of the controller is not supported.

is_unsupported_api_version ()
Whether the API version of communication and controller do not match.

Return type bool
Returns True

class AYABInterface.communication.states. InitializingMachine (communication)
Bases: AYABInterface.communication.states.State

The machine is currently being initialized.

is_before_knitting ()
Whether the knitting should start soon.
class AYABInterface.communication.states.StartingToKnit (communication)
Bases: AYABInterface.communication.states.State

The cnfStart Message is sent and we wait for an answer.

enter ()
Send a StartRequest.

is_before_knitting ()
The knitting should start soon.

return type bool
Returns True

receive_state_indication (message)
Receive a StateIndication message.

Parameters message – a StateIndication message

If the message says that the controller is is ready to knit, there is a transition to StartingToKnit or else the messages are ignored because they come from The reqTest Message.

class AYABInterface.communication.states.StartingFailed (communication)
Bases: AYABInterface.communication.states.FinalState

The starting process has failed.

is_starting_failed ()
The machine machine could not be configured to start.

return type bool
Returns True
class AYABInterface.communication.states.KnittingStarted (communication)
    Bases: AYABInterface.communication.states.State

    The knitting started and we are ready to receive The reqLine Message.

    is_knitting ()
        The machine ready to knit or knitting.
        
        Return type bool
        Returns True

    is_knitting_started()
        The machine ready to knit the first line.
        
        Return type bool
        Returns True

    receive_line_request (message)
        Receive a LineRequest message.
        
        Parameters message – a LineRequest message
        The communication transisitions into a KnittingLine.

class AYABInterface.communication.states.KnittingLine (communication, line_number)
    Bases: AYABInterface.communication.states.State

    The machine is currently knitting a line.

    __init__ (communication, line_number)
        The machine is knitting a line.

    enter()
        Send a LineConfirmation to the controller.
        
        When this state is entered, a AYABInterface.communication.host_messages.LineConfirmation is sent to the controller. Also, the last line requested is set.

    is_knitting()
        The machine ready to knit or knitting.

        Return type bool
        Returns True

    is_knitting_last_line()
        Whether the line currently knit is the last line.

        Return type bool

    is_knitting_line()
        The machine knits a line.

        Return type bool
        Returns True

    line_number
        The number if the line which is currently knit.

        Return type int

    receive_line_request (message)
        Receive a LineRequest message.
Parameters message – a LineRequest message

The communication transitions into a KnittingLine.

class AYABInterface.communication.states.FinalState (communication)
    Bases: AYABInterface.communication.states.State

Base class for states that can not reach knitting.

    is_final ()
    From the current state, the knitting can not be reached.

        Return type  bool
        Returns  True
Communication Specification

This document specifies the communication between the host and a controller with the AYAB firmware.

4.1 Serial Communication

115200 baud
Line Ending: \n\r (10 13) Each message ends with a Line Ending.
4.2 Sequence Chart

- **Initial Handshake**
  - `regInfo`
  - `cnfInfo <versionNumber>`
    - **Sending Firmware Version**

- **End of Handshake**
  - `regStart <start, stop>`
    - **Start knitting with start and stop Needles**
  - `cnfStart <success>`

- **Requests line number nr**
  - `regLine <nr>`
  - `cnfLine <nr, data, crc, last = 0>`
  - `regLine <nr+1>`

- **Confirm the start request with true when is ready.**

- Repeat until last line
The host waits for a `indState(true)` message before requesting to start the knitting. On startup, the Arduino continuously checks for the initialization of the machine (carriage passed left hall sensor). When this happens, it sends an `indState(true)` to tell the host that the machine is ready to knit. After receiving this message, the host sends a `reqStart` message, which is immediately confirmed with a `cnfStart` message. When `reqStart` was successful, the Arduino begins to poll the host for line data with `reqLine`, the host answers with `cnfLine`. This `reqLine/cnfLine` happens each time the carriage moves passed the borders given by the `Start/StopNeedle` parameters in `reqStart`. When the host does not have any more lines to send, it marks the last line with the `lastLine` flag in its last `cnfLine` message.

To see an example implementation, see the states of the communication module.

### 4.3 Message Identifier Format

Messages start with a byte that identifies their type. This byte is called “id” or “message id” in the following document. This table lists all the bits of this byte and assigns their purpose:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Name</th>
<th>Description and Values</th>
</tr>
</thead>
</table>
| 7   | 128   | message source    | • 0 = the message is from the host  
|     |       |                   | • 1 = the message is from the controller |
| 6   | 64    | message type      | • 0 = the message is a request  
|     |       |                   | • 1 = the message is a confirmation of a request |
| 5   | 32    | reserved           | must be zero |
| 4   | 16    |                   | |
| 3   | 8     | message identifier | These are the values that identify the message. |
| 2   | 4     |                   | |
| 1   | 2     |                   | |
| 0   | 1     |                   | |

### 4.4 Message definitions (API v4)

The length is the total length with `id` and parameters. Note that the two characters `\r\n` following the message are not included in the length.

**Message definitions (API v4)**
<table>
<thead>
<tr>
<th>source</th>
<th>name</th>
<th>id</th>
<th>length</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>reqStart</td>
<td>0x01</td>
<td>3</td>
<td>0xaa 0xbb</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• aa = left end needle (Range: 0..198)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• bb = right end needle (Range: 1..199)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Start and</td>
</tr>
<tr>
<td>hardware</td>
<td>cnfStart</td>
<td>0xC1</td>
<td>2</td>
<td>0x0a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• a = success (0 = false, 1 = true)</td>
</tr>
<tr>
<td>hardware</td>
<td>reqLine</td>
<td>0x82</td>
<td>2</td>
<td>0xaa</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• aa = line number (Range: 0..255)</td>
</tr>
<tr>
<td>host</td>
<td>cnfLine</td>
<td>0x42</td>
<td>29</td>
<td>0x0a 0xaa 0xbb[24,23,22,...1,0] 0xcc 0xdd</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• aa = line number (Range: 0..255)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• bb[24 to 0] = binary pixel data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• cc = flags (bit 0: lastLine)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• dd = CRC8 Checksum</td>
</tr>
<tr>
<td>host</td>
<td>reqInfo</td>
<td>0x03</td>
<td>1</td>
<td>0xaa 0xbb 0x0c0x0d</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• a = ready (0 = false, 1 = true)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• BBbb = int left hall sensor value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• CCCcc = int right hall sensor value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• dd = the carriage</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 0 = no carriage</td>
</tr>
<tr>
<td>hardware</td>
<td>cnfInfo</td>
<td>0xC3</td>
<td>4</td>
<td>0xaa 0xbb 0x0c0x0d</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• a = API Version Identifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• bb = Firmware Major Version</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• cc = Firmware Minor Version</td>
</tr>
<tr>
<td>hardware</td>
<td>indState</td>
<td>0x84</td>
<td>8</td>
<td>0x0a 0xBB 0xbb 0x0c0x0d0xee</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• a = ready (0 = false, 1 = true)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• BBbb = int left hall sensor value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• CCCcc = int right hall sensor value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• dd = the carriage</td>
</tr>
</tbody>
</table>

4.4. Message definitions (API v4)
4.4.1 The reqStart Message

The host starts the knitting process.

- Python: `StartRequest`
- Arduino: `h_reqStart`
- table: `reqStart`
- requests answer: The cnfStart Message
- direction: host → controller

4.4.2 The cnfStart Message

The controller indicates the success of The reqStart Message.

- Python: `StartConfirmation`
- Arduino: `h_reqStart`
- table: `reqStart`
- answers: The reqStart Message
- direction: controller → host

4.4.3 The reqLine Message

The controller requests a new line from the host.

More than 256 lines are supported. There are three possibilities for the next line based on the last line:

1. the new line is greater than the last line
2. the new line is lower than the last line
3. the new line is the last line

We choose the line closest to the last line. This is trivial for (3). In case two lines are equally distant from the last line, we choose the smaller line.

This is computed by the function `AYABInterface.utils.next_line()` which is tested and can be seen as a reference implementation for other languages.

- Python: `LineRequest`
- Arduino: `Knitter::reqLine`
- table: `reqLine`
- requests answer: The cnfLine Message
- direction: controller → host

4.4.4 The cnfLine Message

The host answers The reqLine Message with a line configuration. This table shows the message content without the first byte that identifies the message:
<table>
<thead>
<tr>
<th>Byte</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>line number</td>
<td>These are the lowest 8 bit of the line. They must match the line number in</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>The reqLine Message</em>.</td>
</tr>
<tr>
<td>1</td>
<td>needle positions</td>
<td>Each bit of the bytes represents a needle position.</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>flags</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>crc8 checksum</td>
<td></td>
</tr>
</tbody>
</table>

In the following table, you can see the mapping of bytes to needles.

**Note:**

- The **Needles** are counted from the leftmost needle on the machine.
- The **Needle** count starts with 0.
- The **Byte** numbering is taken from the table above.
- The **Bit** numbering is consistent with *Message Identifier Format*. The highest bit has the number 7 and the lowest bit has number 0.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Needle</th>
<th>Bit</th>
<th>Needle</th>
<th>Bit</th>
<th>Needle</th>
<th>Bit</th>
<th>Needle</th>
<th>Bit</th>
<th>Needle</th>
<th>Bit</th>
<th>Needle</th>
<th>Bit</th>
<th>Needle</th>
<th>Bit</th>
<th>Needle</th>
<th>Bit</th>
<th>Needle</th>
<th>Bit</th>
<th>Needle</th>
<th>Bit</th>
<th>Needle</th>
<th>Bit</th>
<th>Needle</th>
<th>Bit</th>
<th>Needle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

- Python: *LineConfirmation*
- Arduino: `h_cnfLine`
- table: `cnfLine`
- answers: *The reqLine Message*
- direction: host → controller

**4.4.5 The reqInfo Message**

The host initializes the handshake.

- Python: *InformationRequest*
- Arduino: `h_reqInfo`
- table: `reqInfo`
- requests answer: *The reqInfo Message*
• direction: host → controller

### 4.4.6 The cnfInfo Message

The controller answers *The reqInfo Message* with the API version.

- **Python:** *InformationConfirmation*
- **Arduino:** `h_reqInfo`
- **table:** `cnfInfo`
- **answers:** *The reqInfo Message*
- **direction:** controller → host

### 4.4.7 The indState Message

This is sent when the controller indicates its state. When *ready* it is

- **1**, then this is the first state indication. The machine is now ready to knit
- **0**, the controller is in test mode. This message is sent periodically. *The reqTest Message* switches this on.

- **Python:** *StateIndication*
- **Arduino:** `Knitter::indState`
- **table:** `indState`
- **direction:** controller → host

### 4.4.8 The debug Message

This message ends with a \r\n like every message. It contains debug information from the controller.

- **Python:** *Debug*
- **Arduino:** `DEBUG_PRINT`
- **table:** `debug`
- **direction:** controller → host

### 4.4.9 The reqTest Message

This message puts the controller in a test mode instead of a knitting mode.

- **Python:** *TestRequest*
- **Arduino:** `h_reqTest`
- **table:** `reqTest`
- **requests answer:** *The cnfTest Message*
- **direction:** host → controller
4.4.10 The cnfTest Message

This message confirms whether the controller is in the test mode. If success is indicated, the controller sends *The indState Message* messages periodically, containing the sensor and position values.

- Python: *TestConfirmation*
- Arduino: *h_reqTest*
- table: *cnfTest*
- answers: *The reqTest Message*
- direction: controller → host

4.4.11 References

See also:

- the original specification
- the hardware messages module for messages sent by the hardware
- the host messages module for messages sent by the host
- a discussion about the specification
Indices and tables

- genindex
- modindex
- search
<table>
<thead>
<tr>
<th>Module</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AYABInterface</td>
<td>9</td>
</tr>
<tr>
<td>AYABInterface.actions</td>
<td>9</td>
</tr>
<tr>
<td>AYABInterface.carriages</td>
<td>13</td>
</tr>
<tr>
<td>AYABInterface.communication</td>
<td>20</td>
</tr>
<tr>
<td>AYABInterface.communication.cache</td>
<td>23</td>
</tr>
<tr>
<td>AYABInterface.communication.carriages</td>
<td>24</td>
</tr>
<tr>
<td>AYABInterface.communication.hardware_messages</td>
<td>25</td>
</tr>
<tr>
<td>AYABInterface.communication.host_messages</td>
<td>31</td>
</tr>
<tr>
<td>AYABInterface.communication.states</td>
<td>34</td>
</tr>
<tr>
<td>AYABInterface.convert</td>
<td>20</td>
</tr>
<tr>
<td>AYABInterface.interaction</td>
<td>13</td>
</tr>
<tr>
<td>AYABInterface.machines</td>
<td>14</td>
</tr>
<tr>
<td>AYABInterface.needle_positions</td>
<td>17</td>
</tr>
<tr>
<td>AYABInterface.serial</td>
<td>18</td>
</tr>
<tr>
<td>AYABInterface.utils</td>
<td>19</td>
</tr>
</tbody>
</table>
Symbols

__eq__() (AYABInterface.actions.Action method), 10
__eq__() (AYABInterface.carriages.Carriage method), 13
__eq__() (AYABInterface.machines.Machine method), 14
__getnewargs__() (AYABInterface.communication.hardware_messages.FirmwareVersion method), 31
__getnewargs__() (AYABInterface.convert.NeedlePositions method), 20
__getstate__() (AYABInterface.communication.hardware_messages.FirmwareVersion method), 31
__getstate__() (AYABInterface.convert.NeedlePositions method), 20
__hash__() (AYABInterface.actions.Action method), 10
__hash__() (AYABInterface.carriages.Carriage method), 13
__hash__() (AYABInterface.machines.Machine method), 14
__init__() (AYABInterface.actions.Action method), 10
__init__() (AYABInterface.actions.ActionMetaClass method), 10
__init__() (AYABInterface.communication.Communication method), 20
__init__() (AYABInterface.communication.cache.NeedlePositionCache method), 23
__init__() (AYABInterface.communication.carriages.Carriage method), 25
__init__() (AYABInterface.communication.hardware_messages.FirmwareVersion method), 31
__init__() (AYABInterface.communication.hardware_messages.Message method), 29
__init__() (AYABInterface.communication.states.State method), 34
__new__() (AYABInterface.communication.states.KnittingLine method), 39
__new__() (AYABInterface.communication.states.State method), 34
__repr__() (AYABInterface.actions.Action method), 10
__repr__() (AYABInterface.carriages.Carriage method), 13
__repr__() (AYABInterface.communication.hardware_messages.FirmwareVersion static method), 31
__repr__() (AYABInterface.convert.NeedlePositions static method), 20
__weakref__() (AYABInterface.actions.Action attribute), 10
__weakref__() (AYABInterface.carriages.Carriage attribute), 13
Index
is_connection_closed() (AYABInterface.communication.hardware_messages.Message)

is_connection_closed() (AYABInterface.communication.states.ConnectionClosed)

is_connection_closed() (AYABInterface.communication.states.State)

is_debug() (AYABInterface.communication.hardware_messages.Debug)

is_debug() (AYABInterface.communication.hardware_messages.Message)

is_final() (AYABInterface.communication.states.FinalState)

is_final() (AYABInterface.communication.states.State)

is_from_controller() (AYABInterface.communication.hardware_messages.Message)

is_from_controller() (AYABInterface.communication.host_messages.Message)

is_from_host() (AYABInterface.communication.hardware_messages.Message)

is_from_host() (AYABInterface.communication.host_messages.Message)

is_hole_carriage() (AYABInterface.communication.carriages.Carriage)

is_hole_carriage() (AYABInterface.communication.carriages.HoleCarriage)

is_information_confirmation() (AYABInterface.communication.hardware_messages.InformationConfirmation)

is_information_confirmation() (AYABInterface.communication.hardware_messages.Message)

is_initial_handshake() (AYABInterface.communication.states.InitialHandshake)

is_initial_handshake() (AYABInterface.communication.states.State)

is_initializing_machine() (AYABInterface.communication.states.InitializingMachine)

is_initializing_machine() (AYABInterface.communication.states.State)

is_kh270() (AYABInterface.machines.Machine)

is_kh900() (AYABInterface.machines.Machine)

is_kh910() (AYABInterface.machines.Machine)

is_kh930() (AYABInterface.machines.Machine)

is_kh950() (AYABInterface.machines.Machine)

is_kh965() (AYABInterface.machines.Machine)

is_knit_carriage() (AYABInterface.communication.carriages.Carriage)

is_knit_carriage() (AYABInterface.communication.carriages.KnitCarriage)

is_knitting() (AYABInterface.communication.states.KnittingLine)

is_knitting() (AYABInterface.communication.states.KnittingStarted)

is_knitting() (AYABInterface.communication.states.State)

is_knitting_last_line() (AYABInterface.communication.states.KnittingLine)

is_knitting_line() (AYABInterface.communication.states.KnittingLine)

is_knitting_last_line() (AYABInterface.communication.states.KnittingLine)

is_knitting_line() (AYABInterface.communication.states.KnittingLine)

is_knitting_started() (AYABInterface.communication.states.KnittingStarted)

is_knitting_started() (AYABInterface.communication.states.State)

is_last() (AYABInterface.communication.cache.NeedlePositionCache)

is_line_request() (AYABInterface.communication.hardware_messages.LineRequest)

is_move_carriage_over_left_hall_sensor() (AYABInterface.actions.MoveCarriageOverLeftHallSensor)

is_move_carriage_over_left_hall_sensor() (AYABInterface.actions.MoveCarriageOverLeftHallSensor)
is_move_carriage_to_the_left() (AYABInterface.actions.MoveCarriageToTheLeft method), 12
is_move_carriage_to_the_right() (AYABInterface.actions.MoveCarriageToTheRight method), 12
is_move_needles_into_position() (AYABInterface.actions.MoveNeedlesIntoPosition method), 10
is_put_color_in_nut_a() (AYABInterface.actions.PutColorInNutA method), 12
is_put_color_in_nut_b() (AYABInterface.actions.PutColorInNutB method), 12
is_ready_to_knit() (AYABInterface.actions.PutColorInNutB method), 12
is_start_confirmation() (AYABInterface.communication.hardware_messages.Message method), 30
is_start_confirmation() (AYABInterface.communication.hardware_messages.StateIndication method), 26
is_start_confirmation() (AYABInterface.communication.states.StartingToKnit method), 38
is_starting_failed() (AYABInterface.communication.states.StartingFailed method), 38
is_starting_to_knit() (AYABInterface.communication.states.StartingToKnit method), 38
is_state_indication() (AYABInterface.communication.hardware_messages.Message method), 30
is_state_indication() (AYABInterface.communication.hardware_messages.StateIndication method), 26
is_success() (AYABInterface.communication.hardware_messages.SuccessConfirmation method), 29
is_valid() (AYABInterface.communication.hardware_messages.Message method), 30
is_valid() (AYABInterface.communication.hardware_messages.StateIndication method), 26
is_valid() (AYABInterface.communication.hardware_messages.SuccessConfirmation method), 29
is_valid() (AYABInterface.communication.hardware_messages.UnknownMessage method), 29
is_waiting_for_carriage_to_pass_the_left_turn_mark() (AYABInterface.communication.states.InitializingMachine method), 38
is_switch_carriage_to_mode_kc() (AYABInterface.actions.SwitchCarriageToModeKc method), 11
is_switch_carriage_to_mode_nl() (AYABInterface.actions.SwitchCarriageToModeNl method), 11
is_switch_off_machine() (AYABInterface.actions.SwitchOffMachine method), 13
is_switch_on_machine() (AYABInterface.actions.SwitchOnMachine method), 13
is_unknown() (AYABInterface.communication.hardware_messages.Message method), 30
is_unknown() (AYABInterface.communication.hardware_messages.StateIndication method), 26
is_unknown() (AYABInterface.communication.carriages.Carriage method), 25
is_unsupported_api_version() (AYABInterface.communication.states.State method), 35
is_unsupported_api_version() (AYABInterface.communication.states.UnsupportedApiVersion method), 37
is_valid() (AYABInterface.communication.hardware_messages.Message method), 30
is_valid() (AYABInterface.communication.hardware_messages.StateIndication method), 26
is_valid() (AYABInterface.communication.hardware_messages.SuccessConfirmation method), 29
is_valid() (AYABInterface.communication.hardware_messages.UnknownMessage method), 29
is_valid() (AYABInterface.communication.states.InitializingMachine method), 38
is_test_confirmation() (AYABInterface.communication.hardware_messages.Message method), 27
is_test_confirmation() (AYABInterface.communication.hardware_messages.TestConfirmation method), 27
is_unknown_carriage() (AYABInterface.communication.states.Carriage method), 25
is_unsupported_api_version() (AYABInterface.communication.statesExpanded.UnsupportedApiVersion method), 37
is_waiting_for_carriage_to_pass_the_left_turn_mark() (AYABInterface.communication.states.InitializingMachine method), 38
is_waiting_for_start() (AYABInterface.communication.states.State method), 35
is_waiting_for_start() (AYABInterface.communication.states.WaitingForStart method), 37
is_waiting_for_the_communication_to_start() (AYABInterface.communication.states.State method), 35

K
KH270 (class in AYABInterface.machines), 17
KH900 (class in AYABInterface.machines), 16
KH910 (class in AYABInterface.machines), 16
KH930 (class in AYABInterface.machines), 16
KH950 (class in AYABInterface.machines), 17
KH965 (class in AYABInterface.machines), 17
KH9XXSeries (class in AYABInterface.machines), 16
KnitCarriage (class in AYABInterface.carriages), 13
KnitCarriage (class in AYABInterface.communication.carriages), 24
KnittingLine (class in AYABInterface.communication.states), 39
KnittingStarted (class in AYABInterface.communication.states), 38

L
last_requested_line_number (AYABInterface.communication.Communication attribute), 21
left_end_needle (AYABInterface.communication.Communication attribute), 22
left_end_needle (AYABInterface.communication.host_messages.StartRequest attribute), 33
left_end_needle (AYABInterface.machines.Machine attribute), 15
left_hall_sensor_value (AYABInterface.communication.hardware_messages.StateIndication attribute), 26
line_number (AYABInterface.communication.hardware_messages.LineRequest attribute), 26
line_number (AYABInterface.communication.states.KnittingLine attribute), 39
LineConfirmation (class in AYABInterface.communication.host_messages), 33
LineRequest (class in AYABInterface.communication.hardware_messages), 26
list_serial_port_strings() (in module AYABInterface.serial), 18
list_serial_ports() (in module AYABInterface.serial), 18
lock (AYABInterface.communication.Communication attribute), 22

M
machine (AYABInterface.needle_positions.NeedlePositions attribute), 18
Machine (class in AYABInterface.machines), 14
major (AYABInterface.communication.hardware_messages.FirmwareVersion attribute), 31
Message (class in AYABInterface.communication.hardware_messages), 29
Message (class in AYABInterface.communication.host_messages), 31
MESSAGE_ID (AYABInterface.communication.hardware_messages.InformationConfirmation attribute), 27
MESSAGE_ID (AYABInterface.communication.hardware_messages.LineRequest attribute), 26
MESSAGE_ID (AYABInterface.communication.hardware_messages.StartConfirmation attribute), 28
MESSAGE_ID (AYABInterface.communication.hardware_messages.StateIndication attribute), 25
MESSAGE_ID (AYABInterface.communication.hardware_messages.TestConfirmation attribute), 27
MESSAGE_ID (AYABInterface.communication.host_messages.InformationRequest attribute), 33
MESSAGE_ID (AYABInterface.communication.host_messages.LineConfirmation attribute), 33
MESSAGE_ID (AYABInterface.communication.host_messages.Message attribute), 31
MESSAGE_ID (AYABInterface.communication.host_messages.StartRequest attribute), 32
MESSAGE_ID (AYABInterface.communication.host_messages.TestRequest attribute), 33
minor (AYABInterface.communication.hardware_messages.FirmwareVersion attribute), 31
MoveCarriageOverLeftHallSensor (class in AYABInterface.actions), 12
MoveCarriageToTheLeft (class in AYABInterface.actions), 12
MoveCarriageToTheRight (class in AYABInterface.actions), 12
MoveNeedlesIntoPosition (class in AYABInterface.actions), 12
NAME (AYABInterface.machines.Machine attribute), 14
name (AYABInterface.machines.Machine attribute), 15
name (AYABInterface.serial.SerialPort attribute), 19
needle_coloring (AYABInterface.convert.NeedlePositions attribute), 20
needle_position (AYABInterface.communication.carriages.Carriage attribute), 25
needle_positions (AYABInterface.communication.Communication attribute), 22
needle_positions (AYABInterface.machines.CK35 attribute), 16
needle_positions (AYABInterface.machines.KH270 attribute), 17
needle_positions (AYABInterface.machines.KH9XXSeries attribute), 16
needle_positions (AYABInterface.machines.Machine attribute), 15
needle_positions_to_bytes() (AYABInterface.machines.Machine method), 15
NeedlePositionCache (class in AYABInterface.communication.cache), 23
NeedlePositions (class in AYABInterface.convert), 20
NeedlePositions (class in AYABInterface.needle_positions), 17
NeedlePositions() (in module AYABInterface), 9
next_line() (in module AYABInterface.utils), 19
NullCarriage (class in AYABInterface.communication.carriages), 24
number_of_colors() (in module AYABInterface.utils), 19
number_of_needles (AYABInterface.machines.CK35 attribute), 16
number_of_needles (AYABInterface.machines.KH270 attribute), 17
number_of_needles (AYABInterface.machines.KH9XXSeries attribute), 16
number_of_needles (AYABInterface.machines.Machine attribute), 16

parallelize() (AYABInterface.communication.Communication method), 22
PutColorInNutA (class in AYABInterface.actions), 12
PutColorInNutB (class in AYABInterface.actions), 12

R
read_end_of_message() (AYABInterface.communication.hardware_messages.FixedSizeMessage method), 31
read_message_type() (in module AYABInterface.communication.hardware_messages), 25
receive_connection_closed() (AYABInterface.communication.states.State method), 35
receive_debug() (AYABInterface.communication.states.State method), 35
receive_information_confirmation() (AYABInterface.communication.states.InitialHandshake method), 37
receive_information_confirmation() (AYABInterface.communication.states.State method), 36
receive_line_request() (AYABInterface.communication.states.KnittingLine method), 39
receive_line_request() (AYABInterface.communication.states.KnittingStarted method), 39
receive_line_request() (AYABInterface.communication.states.State method), 36
receive_message() (AYABInterface.communication.states.State method), 36
receive_message() (AYABInterface.communication.states.State method), 36
receive_start_confirmation() (AYABInterface.communication.states.StartingToKnit method), 38
receive_start_confirmation() (AYABInterface.communication.states.State method), 36
receive_state_indication() (AYABInterface.communication.states.InitializingMachine method), 38
receive_state_indication() (AYABInterface.communication.states.State method), 36
receive_test_confirmation() (AYABInterface.communication.states.State method), 36

Index 61