
Club Robot Documentation

Release 1.0.0

Mathis Lécivain

Sep 21, 2020

CONTENTS:

1	Club Robot API	1
1.1	Class Hierarchy	1
1.2	File Hierarchy	1
1.3	Full API	1
2	Indices and tables	175
	Index	177

CLUB ROBOT API

1.1 Class Hierarchy

1.2 File Hierarchy

1.3 Full API

1.3.1 Namespaces

Namespace IK

Contents

- *Classes*
- *Functions*

Classes

- *Class ArmManager*
- *Class Joint*
- *Class Matrix2*
- *Class Matrix3*
- *Class MotorWrapper*
- *Class Picker*
- *Class Scara*
- *Class TrajectoryManager*

Functions

- *Function IK::equals*
- *Function IK::float_equals*
- *Function IK::operator<<(ostream&, const TrajectoryTime&)*
- *Function IK::operator<<(ostream&, const Coords&)*
- *Template Function IK::operator<<(ostream&, const vector<T>&)*
- *Function IK::operator<<(ostream&, const Joints&)*
- *Function IK::operator<<(ostream&, const DetailedPos&)*
- *Function IK::operator<<(ostream&, const path_t&)*
- *Function IK::operator<<(ostream&, const vector_t&)*

Namespace std

1.3.2 Classes and Structs

Struct Acc

- Defined in file_esp32_common_IK_datatype.h

Struct Documentation

struct Acc

Public Functions

Acc ()

Acc (float *min*, float *max*)

Public Members

float **min**

float **max**

Struct Constraints

- Defined in file_esp32_common_IK_datatype.h

Struct Documentation

struct Constraints

Public Functions

Constraints ()

Constraints (*Pos* *pos*, *Vel* *vel*, *Acc* *acc*)

Public Members

Pos **pos**

Vel **vel**

Acc **acc**

Struct Coords

- Defined in file_esp32_common_IK_datatype.h

Struct Documentation

struct Coords

Public Functions

Coords ()

Coords (float *x*, float *y*, float *phi*)

Public Members

float **x**

float **y**

float **phi**

Struct Deserializer

- Defined in file_arduino_common_serialutils.h

Struct Documentation

struct Deserializer

Objet destiné à extraire des variables d'un flux en octet.

Deserializer permet d'extraire d'un buffer des variables. Cela permet une utilisation plus simple de *SerialTalks*. Voir l'utilisation dans la doc python. Attention a bien extraire les variables dans le bonne ordre pour éviter les problèmes d'encodage et autres.

Public Functions

Deserializer (*byte buffer*[])

Construct a new *Deserializer* object.

Parameters

- *buffer*: The buffer pointer

template<typename **T**>

Deserializer &**operator**>> (*T &object*)

Opérateur de décalage, à utiliser pour extraire les variables du buffer.

Return *Deserializer*& Retourne le pointeur du deserializer pour une utilisation plus simple.

Parameters

- *object*: bject à compléter avec le buffer. Attention le type de la variable est pris en compte dans la conversion octect -> var

Deserializer &**operator**>> (*char *string*)

Opérateur de décalage, a utilisé pour remplir le buffer uniquement pour les variables de type char.

Return *Deserializer*& Retourne le pointeur du serializer pour une utilisation plus simple

Parameters

- *string*: Object a renvoyer dans le buffer pour transmission.

template<typename **T**>

T **read** ()

Methode interne pour convertir les octets du buffer en données exploitables.

Return **T** Retourne la valeur extraite du buffer.

void **clear** ()

Free the buffer memory.

Deserializer (*byte buffer*[])

template<typename **T**>

Deserializer &**operator**>> (*T &object*)

Deserializer &**operator**>> (*char *string*)

template<typename **T**>

T **read** ()

template<>

String **read** ()
Deserializer template to a string buffer.

Return String The string buffer

template<>
String **read** ()

Public Members

byte ***buffer**
pointer vers le buffer à vider

byte ***adr**
buffer address pointer

Struct DetailedPos

- Defined in file_esp32_common_IK_datatype.h

Struct Documentation

struct DetailedPos

Public Functions

DetailedPos ()

DetailedPos (*Coords* origin, *Coords* link1, *Coords* link2, *Coords* tool)

Public Members

Coords **origin**

Coords **link1**

Coords **link2**

Coords **tool**

Struct Joints

- Defined in file_esp32_common_IK_datatype.h

Struct Documentation

struct Joints

Public Functions

Joints ()

Joints (float *th1*, float *th2*, float *th3*)

Public Members

float **th1**

float **th2**

float **th3**

Struct Motor_state_t

- Defined in file_esp32_common_IK_MotorWrapper.h

Struct Documentation

struct Motor_state_t

Public Members

uint8_t **id**

uint8_t **timeout**

uint8_t **err_code**

Struct MoveCommand_t

- Defined in file_esp32_common_IK_MoveBatch.h

Struct Documentation

struct MoveCommand_t

Public Members

bool **isActive**
float **position**
vector<float> **vel**
vector<float> **time**

Struct path_t

- Defined in file_esp32_common_IK_datatype.h

Struct Documentation

struct path_t

Public Members

vector_t **path_th1**
vector_t **path_th2**
vector_t **path_th3**
Coords **pos**

Struct Polynom

- Defined in file_esp32_common_IK_datatype.h

Struct Documentation

struct Polynom

Public Functions

Polynom ()
Polynom (float *a0*, float *a1*, float *a2*)

Public Members

float **a0**
float **a1**
float **a2**

Struct Pos

- Defined in file_esp32_common_IK_datatype.h

Struct Documentation

struct Pos

Public Functions

Pos ()

Pos (float *min*, float *max*)

Public Members

float **min**

float **max**

Struct Position

- Defined in file_arduino_common_Odometry.h

Struct Documentation

struct Position

Structure de position.

Author Ulysse Darmet *Position* est une structure de Odometry.h qui permet en une variable d'obtenir la totalité des informations à propos du positionnement du robot.

Return struct *Position* {

Public Functions

Position ()

Constructeur de *Position*. Constructeur de *Position* qui initialise la position au coordonnées (0,0) et à l'angle 0.

Position (float *x*, float *y*, float *theta*)

Constructeur de *Position*.

Constructeur de *Position* qui initialise la position au coordonnées indiqués.

Parameters

- *x*: coordonnée en x initial.
- *y*: coordonnée en y initial.
- *theta*: angle initial.

Position ()

Position (float *x*, float *y*, float *theta*)

Public Members

float **x**

Coordonnée en x.

float **y**

Coordonnée en y.

float **theta**

Angle.

Struct `PurePursuit::Waypoint`

- Defined in file_arduino_common_PurePursuit.h

Nested Relationships

This struct is a nested type of *Class PurePursuit*.

Struct Documentation

struct *PurePursuit::Waypoint*

Structure d'un point de passage de Purpursuit.

struct *Waypoint*

Public Functions

Waypoint ()

Waypoint (float *x*, float *y*)

Waypoint (const *Position* &*pos*)

Waypoint ()

Waypoint (float *x*, float *y*)

Waypoint (const *Position* &*pos*)

Public Members

float **x**

float **y**

Struct Serializer

- Defined in file_arduino_common_serialutils.h

Struct Documentation

struct Serializer

Objet destiné à créer un flux de sortie pour les programmes C++.

Serializer permet de remplir un buffer en octet à l'aide d'une variable de tous types. Cela permet une utilisation plus simple de *SerialTalks*. Voir l'utilisation dans la doc Python.

Public Functions

Serializer (*byte buffer*[])

Construct a new *Serializer* object.

Parameters

- *buffer*: pointeur du buffer.

template<typename T>

Serializer &operator<< (const T &*object*)

Opérateur de décalage, utilisé pour remplir le buffer.

Return *Serializer*& Retourne le pointeur du serializer pour une utilisation plus simple

Parameters

- *object*: Objet à renvoyer dans le buffer pour transmission.

template<typename T>

void **write** (const T &*object*)

Méthode pour une utilisation interne qui permet d'écrire sur le buffer après conversion en octets.

Parameters

- *object*: Objet à renvoyer dans le buffer pour écriture.

void **write** (const char **string*)

Méthode pour une utilisation interne qui permet d'écrire sur le buffer après conversion en octets.

Parameters

- *string*: Object (sous la forme d'un char) à renvoyer dans le buffer pour écriture.

Serializer (*byte buffer*[])

template<typename T>

Serializer &operator<< (const T &*object*)

template<typename T>

void **write** (const T &*object*)

void **write** (const char **string*)

template<>

void **write** (const *String* &string)
Serializer template to a string buffer.

Parameters

- string: The string buffer

template<>
void **write** (const *String* &string)

Public Members

byte ***buffer**
pointer vers le buffer à compléter

Struct SerialTopics::subscription_t

- Defined in file_arduino_common_SerialTopics.h

Nested Relationships

This struct is a nested type of *Class SerialTopics*.

Struct Documentation

struct *SerialTopics::subscription_t*
Subscription context structure.

Public Members

Subscription **func**

long **timestep**

long **lasttime**

bool **enable**

Struct TrajectoryTime

- Defined in file_esp32_common_IK_datatype.h

Struct Documentation

struct `TrajectoryTime`

Public Functions

TrajectoryTime ()

TrajectoryTime (float *t1*, float *t2*, float *tf*)

Public Members

float **t1**

float **t2**

float **tf**

Struct `vector_t`

- Defined in `file_esp32_common_IK_datatype.h`

Struct Documentation

struct `vector_t`

Public Members

vector<float> **t**

vector<float> **pos**

vector<float> **vel**

vector<float> **acc**

Struct `Vel`

- Defined in `file_esp32_common_IK_datatype.h`

Struct Documentation

struct `Vel`

Public Functions

Vel ()

Vel (float *min*, float *max*)

Public Members

float **min**

float **max**

Struct Workspace

- Defined in file_esp32_common_IK_datatype.h

Struct Documentation

struct **Workspace**

Public Functions

Workspace ()

Workspace (float *x_min*, float *x_max*, float *y_min*, float *y_max*, int *elbow_or*)

Public Members

float **x_min**

float **x_max**

float **y_min**

float **y_max**

float **elbow_or**

Class AbstractCodewheel

- Defined in file_arduino_common_Odometry.h

Inheritance Relationships

Derived Type

- public **Codewheel** (*Class Codewheel*)

Class Documentation

class `AbstractCodewheel`

Classe abstraite d'une roue codeuse.

Cette classe est à implémenter pour être compatible avec la classe *Odometry*.

Author Ulysse Darmet

Subclassed by *Codewheel*

Public Functions

~AbstractCodewheel ()

float **getTraveledDistance** () = 0

Calcul la distance parcourue.

Méthode à implémenter, dont le rôle est de retourner la distance parcourue depuis le dernier appel de cette méthode (ou depuis l'initialisation de l'objet).

Return Distance parcourue depuis le dernier appel.

float **restart** () = 0

Réinitialise l'objet.

Réinitialise l'objet. C'est à dire, repasse tous les paramètres à leur état initial. De plus cette méthode doit retourner la distance parcourue depuis le dernier appel de `AbstractCodewheel::getTraveledDistance`.

Return Distance parcourue depuis le dernier getter de distance.

~AbstractCodewheel ()

float **getTraveledDistance** () = 0

float **restart** () = 0

Class `AbstractMotor`

- Defined in file_arduino_common_DifferentialController.h

Inheritance Relationships

Derived Type

- public `DCMotor` (*Class* *DCMotor*)

Class Documentation

class **AbstractMotor**

Instance de moteur.

Author Ulysse Darmet Instance de Moteur permettant une parfaite compatibilité entre les classes Motor et le *DifferentialController*.

Subclassed by *DCMotor*

Public Functions

~AbstractMotor ()

Constructeur d'*AbstractMotor*.

Méthode à implémenter.

void **setVelocity** (float *velocity*) = 0

Charge une nouvelle vitesse.

Change la vitesse du moteur par celle passée en parametre. Méthode à implémenter.

Parameters

- *velocity*: Nouvelle vitesse.

float **getMaxVelocity** () **const** = 0

Retourne vitesse max.

Retourne la vitesse max du moteur pour son intégration dans les calculs de *DifferentialController*.

Return virtual float

~AbstractMotor ()

void **setVelocity** (float *velocity*) = 0

float **getMaxVelocity** () **const** = 0

Class **AbstractMoveStrategy**

- Defined in file_arduino_common_PositionController.h

Inheritance Relationships

Derived Types

- public *PurePursuit* (Class *PurePursuit*)
- public *TurnOnTheSpot* (Class *TurnOnTheSpot*)

Class Documentation

class **AbstractMoveStrategy**

Interface de Stratégie de mouvement.

Interface à implémenter pour réaliser une classe de strategie de mouvement.

Subclassed by *PurePursuit*, *TurnOnTheSpot*

Protected Functions

void **computeVelSetpoints** (float *timestep*) = 0

Calcul les nouvelles vitesses désirer.

Méthode à implémenter pour réaliser une *AbstractMoveStrategy*. Cette méthode calcul à partir de la position du robot des vitesses à suivre pour le robot.

Parameters

- *timestep*: Temps depuis le dernier appel en secondes.

bool **getPositionReached** () = 0

Indique si la position désirée est atteinte.

Calcul la distance entre la position du robot et la position désirée selon le mode de calcul de l'*AbstractMoveStrategy*.

Return true Si la position est atteinte.

Return false Si la position n'est pas atteinte.

const *Position* &**getPosInput** () const

Retourne la position du robot.

Retourne la position du robot stocker dans le *PositionController*.

Return La position du robot sous la struct *Position*.

const *Position* &**getPosSetpoint** () const

Retourne la position à atteindre.

Return *Position* à atteindre.

void **setVelSetpoints** (float *linVelSetpoint*, float *angVelSetpoint*)

Charge une nouvelle vitesse pour le robot.

Parameters

- *linVelSetpoint*: Vitesse linéaire en mm/s.
- *angVelSetpoint*: Vitesse angulaire en rad/s.

float **getLinVelKp** () const

Retourne le coef proportionnel de vitesse linéaire.

Return Coefficient proportionnel (sans unité).

float **getAngVelKp** () **const**
 Retourne le coef proportionnel de vitesse angulaire.

Return Coefficient proportionnel (sans unité).

float **getLinVelMax** () **const**
 Retourne vitesse linéaire max.

Return Vitesse en mm/s.

float **getAngVelMax** () **const**
 Retourne vitesse angulaire max.

Return Vitesse en rad/s.

float **getLinPosThreshold** () **const**
 Retourne la précision cartésienne à atteindre.

Return Précision en mm.

float **getAngPosThreshold** () **const**
 Retourne la précision angulaire à atteindre.

Return Précision en rad.

void **computeVelSetpoints** (float *timestep*) = 0

bool **getPositionReached** () = 0

const *Position* &**getPosInput** () **const**

const *Position* &**getPosSetpoint** () **const**

void **setVelSetpoints** (float *linVelSetpoint*, float *angVelSetpoint*)

float **getLinVelKp** () **const**

float **getAngVelKp** () **const**

float **getLinVelMax** () **const**

float **getAngVelMax** () **const**

float **getLinPosThreshold** () **const**

float **getAngPosThreshold** () **const**

Protected Attributes

PositionController ***m_context**
 Pointeur du PositionControlleur associé.

Friends

friend class PositionController

Class AX12

- Defined in file_arduino_common_AX12_AX12.h

Class Documentation

class AX12

Public Functions

void **attach** (unsigned char *id*)

void **detach** ()

int **ping** ()

int **setID** (unsigned char *newID*)

int **setBD** (long *baud*)

int **move** (float *Position*)

int **moveSpeed** (float *Position*, float *Speed*)

int **setEndlessMode** (bool *Status*)

int **turn** (int *Speed*)

int **Nextmove** (float *Position*)

int **NextmoveSpeed** (float *Position*, float *Speed*)

int **setTempLimit** (unsigned char *Temperature*)

int **setAngleLimit** (float *CWLimit*, float *CCWLimit*)

int **setVoltageLimit** (unsigned char *DVoltage*, unsigned char *UVoltage*)

int **setMaxTorque** (int *MaxTorque*)

int **setMaxTorqueRAM** (int *MaxTorque*)

int **setSRL** (unsigned char *SRL*)

int **setRDT** (unsigned char *RDT*)

int **setLEDAlarm** (unsigned char *LEDAlarm*)

int **setShutdownAlarm** (unsigned char *SALARM*)

int **setCMargin** (unsigned char *CWCMargin*, unsigned char *CCWCMargin*)

int **setCSlope** (unsigned char *CWCSlope*, unsigned char *CCWCSlope*)

int **setPunch** (int *Punch*)

int **moving** ()

int **lockRegister** ()

```
int savedMove ()
int readTemperature ()
float readVoltage ()
float readPosition ()
float readSpeed ()
int readTorque ()
bool isHolding ()
int hold (bool Status)
int led (bool Status)
void attach (unsigned char id)
void detach ()
int reset ()
int ping ()
int setID (unsigned char newID)
int setBD (long baud)
int move (float Position)
int moveSpeed (float Position, float Speed)
int setEndlessMode (bool Status)
int turn (int Speed)
int Nextmove (float Position)
int NextmoveSpeed (float Position, float Speed)
int setTempLimit (unsigned char Temperature)
int setAngleLimit (float CWLimit, float CCWLimit)
int setVoltageLimit (unsigned char DVoltage, unsigned char UVoltage)
int setMaxTorque (int MaxTorque)
int setMaxTorqueRAM (int MaxTorque)
int setSRL (unsigned char SRL)
int setRDT (unsigned char RDT)
int setLEDAlarm (unsigned char LEDAlarm)
int setShutdownAlarm (unsigned char SALARM)
int setCMargin (unsigned char CWCMargin, unsigned char CCWCMargin)
int setCSlope (unsigned char CWCSlope, unsigned char CCWCSlope)
int setPunch (int Punch)
int moving ()
int lockRegister ()
int savedMove ()
```

```
int readTemperature ()
float readVoltage ()
float readPosition ()
float readSpeed ()
int readTorque ()
bool isHolding ()
int hold (bool Status)
int led (bool Status)
```

Public Static Functions

```
void SerialBegin (long baud, unsigned char rx, unsigned char tx, unsigned char control)
void end ()
void action ()
void SerialBegin (long baud, unsigned char control)
void end ()
void action ()
```

Class AX12error

- Defined in file_esp32_common_AX12_Dynamixel.h

Class Documentation

```
class AX12error
```

Public Functions

```
AX12error (int ID, int error_code)
bool resolve_AX_error ()
int get_id () const
int get_error_code () const
```


Class AX12Timeout

- Defined in file_esp32_common_AX12_Dynamixel.h

Class Documentation

```
class AX12Timeout
```

Public Functions

```
AX12Timeout (int id)
```

```
int get_id() const
```

Class BrushlessMotor

- Defined in file_arduino_common_BrushlessMotor.h

Inheritance Relationships

Base Type

- public PeriodicProcess (*Class PeriodicProcess*)

Class Documentation

```
class BrushlessMotor : public PeriodicProcess
```

Public Functions

```
BrushlessMotor ()
```

```
void attach (int PIN)
```

```
void detach ()
```

```
void enableStartup ()
```

```
void disableStartup ()
```

```
void updateStartup ()
```

```
void enableMotor ()
```

```
void disableMotor ()
```

```
int setVelocity (int velocity)
```

```
int setPulsewidth (int pulsewidth)
```

```
void forcePulsewidth (int pulsewidth)
```

```
void update ()
```

```
void startupProcess ()
```

```
float getVelocity () const  
bool isEnabled () const  
int readMicroseconds ()
```

Protected Functions

```
void process (float timestep)
```

Méthode à implémenter obligatoirement pour hériter de *PeriodicProcess*.

Process est la méthode qui s'exécutera toutes les *m_timestep*. Il doit donc définir l'action répétitive voulue dans la loop de l'Arduino.

Parameters

- *timestep*: Temps écoulé depuis le dernier appel en seconde.

Class Clock

- Defined in file_arduino_common_Clock.h

Class Documentation

class Clock

Utilitaire pour gérer le temps dans vos programmes Arduino.

class *Clock*

Author Ulysse Darmet est un outil permettant à vos programmes d'utiliser plus simplement la méthode `micros()` de `<Arduino.h>`. Cette objet vous permettra de mesurer le temps écoulé depuis le dernier appel de la méthode `Clock::restart`.

Public Functions

```
Clock ()
```

Constructeur de *Clock*.

Le constructeur de *Clock* en plus de construire l'objet fait un premier marqueur qui vous permettra d'utiliser `Clock::getElapsedTime` pour avoir le temps écoulé depuis la création de l'objet.

```
float getElapsedTime () const
```

Récupère le temps depuis le dernier reset.

Récupère le temps écoulé en secondes depuis la construction de l'objet ou depuis le dernier `Clock::restart`.

Return Temps écoulé en secondes.

```
float restart ()
```

Reset le temps.

Réinitialise le temps à 0s.

Return Temps écoulé en secondes depuis le dernier reset.

```
Clock ()
```

```
float getElapsedTime () const
```

float **restart** ()

Class Codewheel

- Defined in file_arduino_common_Codewheel.h

Inheritance Relationships

Base Types

- private NonCopyable (*Class NonCopyable*)
- public AbstractCodewheel (*Class AbstractCodewheel*)

Class Documentation

class Codewheel : private *NonCopyable*, public *AbstractCodewheel*

Fait la passerelle entre les roues codeuses et le compteur.

class *Codewheel*

Author Ulysse Darmet Cette classe permet de récupérer les tics des roues codeuses à travers la puce compteuse.

Public Functions

Codewheel ()

void **attachCounter** (int *XY*, int *AXIS*, int *SEL1*, int *SEL2*, int *OE*, int *RST*)

Set les pins pour le compteur.

Parameters

- XY: Voir doc elec
- AXIS: Voir doc elec
- SEL1: Voir doc elec
- SEL2: Voir doc elec
- OE: Voir doc elec
- RST: Voir doc elec

void **attachRegister** (int *DATA*, int *LATCH*, int *CLOCK*)

Set les pins pour les registres du compteur.

Parameters

- DATA:
- LATCH:
- CLOCK:

long **getCounter** ()

Donne le nombre de tic absolue courant.

Return long Le nombre de tics depuis le dernier reset.

long **getCountsPerRev** ()

Donne le nombre de tics par tour courant.

Return long Nombre de tics par tour.

float **getWheelRadius** ()

Donne le rayon de la roue en mm.

Return float rayon en mm.

void **setCountsPerRev** (long *countsPerRev*)

Set le nombre de tics par tour.

Parameters

- *countsPerRev*: nb de tics par tour.

void **setWheelRadius** (float *wheelRadius*)

Set le rayon en mm de la roue.

Parameters

- *wheelRadius*: rayon en mm.

void **reset** ()

Réinitialise le compteur à 0.

Cette méthode peut rendre caduque le prochain `Codewheel::getTraveledDistance`.

float **getTraveledDistance** ()

Donne la distance parcouru.

Cette méthode donne la distance parcouru par la roue depuis le dernier `Codewheel::reset` ou `Codewheel::restart`.

Return float

float **restart** ()

Donne la distance parcouru et reset le compteur.

Cette méthode appelle la méthode `Codewheel::getTraveledDistance` avant de mettre à jour le compteur de l'arduino (pas du compteur (puce elec)).

Return float

void **load** (int *address*)

Charge les données de l'EEPROM avec l'offset.

Parameters

- *address*: Offset à utiliser pour lire dans l'EEPROM.

void **save** (int *address*)

Sauvegarde les constantes actuelles dans l'EEPROM.

Parameters

- `address`: Offset à utiliser pour écrire dans l'EEPROM.

Codewheel ()

void **attachCounter** (int *XY*, int *AXIS*, int *SEL1*, int *SEL2*, int *OE*, int *RST*)
Set les pins pour le compteur.

Parameters

- *XY*: Voir doc elec
- *AXIS*: Voir doc elec
- *SEL1*: Voir doc elec
- *SEL2*: Voir doc elec
- *OE*: Voir doc elec
- *RST*: Voir doc elec

void **attachRegister** (int *DATA*, int *LATCH*, int *CLOCK*)
Set les pins pour les registres du compteur.

Parameters

- *DATA*:
- *LATCH*:
- *CLOCK*:

long **getCounter** ()
Donne le nombre de tics absolue courant.

Return long Le nombre de tics depuis le dernier reset.

long **getCountsPerRev** ()
Donne le nombre de tics par tour courant.

Return long Nombre de tics par tour.

float **getWheelRadius** ()
Donne le rayon de la roue en mm.

Return float rayon en mm.

void **setCountsPerRev** (long *countsPerRev*)
Set le nombre de tics par tour.

Parameters

- *countsPerRev*: nb de tics par tour.

void **setWheelRadius** (float *wheelRadius*)
Set le rayon en mm de la roue.

Parameters

- *wheelRadius*: rayon en mm.

void **reset** ()

Réinitialise le compteur à 0.

Cette méthode peut rendre caduque le prochain `Codewheel::getTraveledDistance`.

float **getTraveledDistance** ()

Donne la distance parcouru.

Cette méthode donne la distance parcouru par la roue depuis le dernier `Codewheel::reset` ou `Codewheel::restart`.

Return float

float **restart** ()

Donne la distance parcouru et reset le compteur.

Cette méthode appelle la méthode `Codewheel::getTraveledDistance` avant de mettre à jour le compteur de l'arduino (pas du compteur (puce elec)).

Return float

void **load** (int *address*)

Charge les données de l'EEPROM avec l'offset.

Parameters

- *address*: Offset à utiliser pour lire dans l'EEPROM.

void **save** (int *address*) **const**

Sauvegarde les constantes actuelles dans l'EEPROM.

Parameters

- *address*: Offset à utiliser pour écrire dans l'EEPROM.

Protected Functions

void **update** ()

Récupère le nombre de tics stockés dans le compteur.

void **update** ()

Récupère le nombre de tics stockés dans le compteur.

Protected Attributes

long **m_currentCounter**

Tic courant.

long **m_startCounter**

Tic depuis le dernier reset.

float **m_wheelRadius**

Rayon de la roue codeuse en mm.

long **m_countsPerRev**

Nombre de tics par tour de roue.

int **m_COUNTER_XY**

Select one of the two quad counters. See below.

```
int m_COUNTER_AXIS
    Not a pin: X = 0, Y = 0.

int m_COUNTER_SEL1
    MSB = 0, 2ND = 1, 3RD = 0, LSB = 1.

int m_COUNTER_SEL2
    MSB = 0, 2ND = 0, 3RD = 1, LSB = 1.

int m_COUNTER_OE
    Active LOW. Enable the tri-states output buffers.

int m_COUNTER_RST
    Active LOW. Clear the internal position counter and the position latch.

int m_REGISTER_DATA
    Serial data input from the 74HC165 register.

int m_REGISTER_LATCH
    Active LOW. Latch signal for the 74HC165 register.

int m_REGISTER_CLOCK
    LOW-to-HIGH edge-triggered. Clock signal for the 74HC165 register.
```

Class CRC16

- Defined in file_arduino_common_CRC16.h

Class Documentation

```
class CRC16
```

Public Functions

```
CRC16 ()

uint16_t CRCprocessByte (uint8_t data)

uint16_t CRCprocessBuffer (const uint8_t *data_p, int length)

bool CRCcheck (uint8_t *data_p, uint16_t length, uint16_t crc)

CRC16 ()

uint16_t CRCprocessByte (uint8_t data)

uint16_t CRCprocessBuffer (const uint8_t *data_p, uint16_t length)

bool CRCcheck (uint8_t *data_p, uint16_t length, uint16_t crc)
```

Class DCMotor

- Defined in file_arduino_common_DCMotor.h

Inheritance Relationships

Base Types

- private NonCopyable (*Class NonCopyable*)
- public AbstractMotor (*Class AbstractMotor*)

Class Documentation

class DCMotor : private *NonCopyable*, public *AbstractMotor*

Pilotage de moteur continu.

class *DCMotor* Remarque : Pour les moteurs qui ne sont reliées à des roues seulement régler la constante du moteur à 1/tension_PWM. Ainsi, on peut simplement contrôler le moteur via setVelocity() en envoyant comme commande la tension de PWM souhaitée.

Author Ulysse Darmet Cette classe permet de contrôler un moteur à courant continu par PWM via un driver Moteur

Public Functions

DCMotor ()

void **attach** (int *EN*, int *PWM*, int *DIR*)

Indique quels pins de l'arduino sont utilisés pour ce moteur (actuellement correspond à moteur 1 ou 2)

void **setVelocity** (float *velocity*)

Envoie une commande de vitesse au moteur.

Parameters

- *velocity*: vitesse de commande en mm/s

void **setConstant** (float *constant*)

Paramètre la constante du moteur.

Parameters

- *constant*: constante en rad/s/Volt

void **setWheelRadius** (float *wheelRadius*)

Paramètre le rayon de la roue liée au moteur.

Parameters

- *wheelRadius*: rayon en mm

void **setMaxPWM** (float *maxPWM*)

Paramètre une valeur limite de PWM à ne pas dépasser.

Parameters

- `maxPWM`: valeur limite entre 0 et 1

void **enable** ()

void **disable** ()

float **getVelocity** () **const**

Renvoie la vitesse de commande actuelle du moteur.

Return vitesse en mm/s

float **getConstant** () **const**

Renvoie la constante du moteur paramétrée.

Return constante : $(60 * \text{reduction_ratio} / \text{velocity_constant_in_RPM}) / \text{supplied_voltage_in_V}$

float **getWheelRadius** () **const**

Renvoie rayon de la roue du moteur.

float **getMaxPWM** () **const**

Renvoie la valeur max de PWM.

Return valeur entre 0 et 1

bool **isEnabled** () **const**

float **getMaxVelocity** () **const**

Renvoie la vitesse maximale avec les constantes actuelles.

Return vitesse max en mm/s

void **load** (int *address*)

void **save** (int *address*) **const**

DCMotor ()

void **attach** (int *EN*, int *PWM*, int *PWMChanel*, int *freq*, int *DIR*)

Indique quels pins de l'arduino son utilisé pour ce moteur (actuellement correspond à moteur 1 ou 2)

void **setVelocity** (float *velocity*)

Envoie une commande de vitesse au moteur.

Parameters

- `velocity`: vitesse de commande en mm/s

void **setConstant** (float *constant*)

Paramètre la constante du moteur.

Parameters

- `constant`: constante en rad/s/Volt

void **setWheelRadius** (float *wheelRadius*)

Paramètre le rayon de la roue liée au moteur.

Parameters

- `wheelRadius`: rayon en mm

void **setMaxPWM** (float *maxPWM*)
Paramètre une valeur limite de PWN à ne pas dépasser.

Parameters

- `maxPWM`: valeur limite entre 0 et 1

void **enable** ()

void **disable** ()

float **getVelocity** () **const**
Renvoie la vitesse de commande actuelle du moteur.

Return vitesse en mm/s

float **getConstant** () **const**
Renvoie la constante du moteur paramétrée.

Return constante : $(60 * \text{reduction_ratio} / \text{velocity_constant_in_RPM}) / \text{supplied_voltage_in_V}$

float **getWheelRadius** () **const**
Renvoie rayon de la roue du moteur.

float **getMaxPWM** () **const**
Renvoie la valeur max de PWM.

Return valeur entre 0 et 1

bool **isEnabled** () **const**

float **getMaxVelocity** () **const**
Renvoie la vitesse maximale avec les constantes actuelles.

Return vitesse max en mm/s

void **load** (int *address*)

void **save** (int *address*) **const**

Protected Functions

void **update** ()

void **update** ()

Protected Attributes

bool **m_enabled**

float **m_velocity**
in mm/s (millimeters per second)

float **m_wheelRadius**
in mm

float **m_constant**
(60 * reduction_ratio / velocity_constant_in_RPM) / supplied_voltage_in_V

float **m_maxPWM**
in range]0, 1]

int **m_EN**

int **m_PWM**

int **m_DIR**

Mutex **m_mutex**

int **m_PWMChanel**

Class DCMotorsDriver

- Defined in file_arduino_common_DCMotor.h

Class Documentation

class DCMotorsDriver

Utilisation des drivers moteurs.

class *DCMotorsDriver*

Author Ulysse Darmet est une classe permettant d'utiliser les fonctions du driver

Public Functions

void **attach** (int *RESET*, int *FAULT*)
Définit les pins utiles au driver.

Parameters

- *RESET*: pin de reset
- *FAULT*: pin de Fault

void **reset** ()

bool **isFaulty** ()

void **attach** (int *RESET*, int *FAULT*)
Définit les pins utiles au driver.

Parameters

- RESET: pin de reset
- FAULT: pin de Fault

```
void reset ()  
bool isFaulty ()
```

Class DifferentialController

- Defined in file_arduino_common_DifferentialController.h

Inheritance Relationships

Base Type

- public `PeriodicProcess` (*Class PeriodicProcess*)

Derived Type

- public `VelocityController` (*Class VelocityController*)

Class Documentation

class DifferentialController : public *PeriodicProcess*

Controle les moteurs.

DifferentialController permet de controler les deux moteurs du robot à partir de l'odométrie et d'un *PID*.

Subclassed by *VelocityController*

Public Functions

DifferentialController ()

Constructeur de *DifferentialController* Constructeur de *DifferentialController* qui initialise les variables à des valeurs neutres.

void **setInputs** (float *linInput*, float *angInput*)

Charge les vitesses actuel.

Charge les vitesses instantanées du robot pour l'asservissement.

Parameters

- *linInput*: Vitesse linéaire en mm/s.
- *angInput*: Vitesse angulaire en rad/s.

void **setSetpoints** (float *linSetpoint*, float *angSetpoint*)

Charge les vitesses désirées.

Charge la vitesse désirée par l'utilisation.

Parameters

- `linSetpoint`: Vitesse linéaire en mm/s.
- `angSetpoint`: Vitesse angulaire en rad/s.

void **setAxleTrack** (float *axleTrack*)

Charge l'entraxe.

Charge l'entraxe entre les deux roues du robot. Attention: cette entraxe est différente que celle du de l'odométrie.

Parameters

- `axleTrack`:

void **setWheels** (*AbstractMotor* &*leftWheel*, *AbstractMotor* &*rightWheel*)

Charge les moteurs.

Charge les pointeurs de *AbstractMotor* du *DifferentialController*.

Parameters

- `leftWheel`: Roue gauche (*AbstractMotor*).
- `rightWheel`: Roue droite (*AbstractMotor*).

void **setPID** (*PID* &*linPID*, *PID* &*angPID*)

Charge l'asservissement.

Charge les pointeurs *PID* pour l'asservissement de *DifferentialController*.

Parameters

- `linPID`: Asservissement linéaire.
- `angPID`: Asservissement angulaire.

float **getLinSetpoint** () **const**

Retourne la vitesse demandée.

Return float Vitesse linéaire en mm/s.

float **getAngSetpoint** () **const**

Retourne la vitesse demandée.

Return float Vitesse angulaire en rad/s.

float **getLinOutput** () **const**

Retour la commande linéaire actuel.

Return float Commande linéaire en mm/s

float **getAngOutput** () **const**

Retour la commande angulaire actuel.

Return float Commande angulaire en rad/s

float **getAxleTrack** () **const**

Retourne l'entraxe.

Return Entraxe en mm.

void **load** (int *address*)
Charge les paramètres.
Charge les paramètres depuis la mémoire de l'Arduino.

Parameters

- *address*: Adresse à utiliser.

void **save** (int *address*) **const**
Sauvegarde les paramètres.
Sauvegarde les paramètres dans la mémoire de l'Arduino.

Parameters

- *address*: Adresse à utiliser.

DifferentialController ()
void **setInputs** (float *linInput*, float *angInput*)
void **setSetpoints** (float *linSetpoint*, float *angSetpoint*)
void **setAxleTrack** (float *axleTrack*)
void **setWheels** (*AbstractMotor* &*leftWheel*, *AbstractMotor* &*rightWheel*)
void **setPID** (*PID* &*linPID*, *PID* &*angPID*)
float **getLinSetpoint** () **const**
float **getAngSetpoint** () **const**
float **getLinOutput** () **const**
float **getAngOutput** () **const**
float **getAxleTrack** () **const**
void **load** (int *address*)
void **save** (int *address*) **const**

Protected Functions

void **process** (float *timestep*)
Calcul l'asservissement

Parameters

- *timestep*: Temps depuis le dernier appel.

void **onProcessEnabling** ()
Reset les accumulateurs des asserv.

void **process** (float *timestep*)
Méthode à implémenter obligatoirement pour hériter de *PeriodicProcess*.

Process est la méthode qui s'exécutera toutes les *m_timestep*. Il doit donc définir l'action répétitive voulue dans la loop de l'Arduino.

Parameters

- `timestep`: Temps écoulé depuis le dernier appel en seconde.

void **onProcessEnabling** ()

Méthode exécutée à l'activation du *PeriodicProcess*.

Méthode à implémenter si votre class nécessite des actions à son activation.

Protected Attributes

float **m_linInput**

Vitesse linéaire actuel du robot.

float **m_angInput**

Vitesse angulaire actuel du robot.

float **m_linSetpoint**

Vitesse linéaire demandée (en mm/s).

float **m_angSetpoint**

Vitesse angulaire demandée (en rad/s).

float **m_axleTrack**

Entraxe entre les deux roues motrices du robot (en mm).

float **m_linVelOutput**

Vitesse linéaire asservie.

float **m_angVelOutput**

Vitesse angulaire asservie.

AbstractMotor ***m_leftWheel**

Pointeur du moteur gauche (*AbstractMotor*).

AbstractMotor ***m_rightWheel**

Pointeur du moteur droit (*AbstractMotor*).

PID ***m_linPID**

Pointeur de l'asservissement linéaire.

PID ***m_angPID**

Pointeur de l'asservissement angulaire.

Mutex **m_mutex**

Class DynamixelClass

- Defined in file_arduino_common_AX12_Dynamixel.h

Class Documentation

class DynamixelClass

Public Functions

void **begin** (long *baud*, unsigned char *Rx*, unsigned char *Tx*)
void **begin** (long *baud*, unsigned char *Rx*, unsigned char *Tx*, unsigned char *D_Pin*)
void **end** (void)
int **reset** (unsigned char *ID*)
int **ping** (unsigned char *ID*)
int **setID** (unsigned char *ID*, unsigned char *newID*)
int **setBD** (unsigned char *ID*, long *baud*)
int **move** (unsigned char *ID*, int *Position*)
int **moveSpeed** (unsigned char *ID*, int *Position*, int *Speed*)
int **setEndless** (unsigned char *ID*, bool *Status*)
int **turn** (unsigned char *ID*, bool *SIDE*, int *Speed*)
int **moveRW** (unsigned char *ID*, int *Position*)
int **moveSpeedRW** (unsigned char *ID*, int *Position*, int *Speed*)
void **action** (void)
int **setTempLimit** (unsigned char *ID*, unsigned char *Temperature*)
int **setAngleLimit** (unsigned char *ID*, int *CWLimit*, int *CCWLimit*)
int **setVoltageLimit** (unsigned char *ID*, unsigned char *DVoltage*, unsigned char *UVoltage*)
int **setMaxTorque** (unsigned char *ID*, int *MaxTorque*)
int **setMaxTorqueRAM** (unsigned char *ID*, int *MaxTorque*)
int **setSRL** (unsigned char *ID*, unsigned char *SRL*)
int **setRDT** (unsigned char *ID*, unsigned char *RDT*)
int **setLEDAlarm** (unsigned char *ID*, unsigned char *LEDAlarm*)
int **setShutdownAlarm** (unsigned char *ID*, unsigned char *SALARM*)
int **setCMargin** (unsigned char *ID*, unsigned char *CWCMargin*, unsigned char *CCWCMargin*)
int **setCSlope** (unsigned char *ID*, unsigned char *CWCSlope*, unsigned char *CCWCSlope*)
int **setPunch** (unsigned char *ID*, int *Punch*)
int **moving** (unsigned char *ID*)
int **lockRegister** (unsigned char *ID*)
int **RWStatus** (unsigned char *ID*)
int **readTemperature** (unsigned char *ID*)
int **readVoltage** (unsigned char *ID*)


```

int readPosition (unsigned char ID)
int readSpeed (unsigned char ID)
int readLoad (unsigned char ID)
int torqueStatus (unsigned char ID, bool Status)
int ledStatus (unsigned char ID, bool Status)
void begin (long baud)
void begin (long baud, unsigned char D_Pin)
void end (void)
int reset (unsigned char ID)
int ping (unsigned char ID)
int setID (unsigned char ID, unsigned char newID)
int setBD (unsigned char ID, long baud)
int move (unsigned char ID, int Position)
int moveSpeed (unsigned char ID, int Position, int Speed)
int setEndless (unsigned char ID, bool Status)
int turn (unsigned char ID, bool SIDE, int Speed)
int moveRW (unsigned char ID, int Position)
int moveSpeedRW (unsigned char ID, int Position, int Speed)
void action (void)
int setTempLimit (unsigned char ID, unsigned char Temperature)
int setAngleLimit (unsigned char ID, int CWLimit, int CCWLimit)
int setVoltageLimit (unsigned char ID, unsigned char DVoltage, unsigned char UVoltage)
int setMaxTorque (unsigned char ID, int MaxTorque)
int setMaxTorqueRAM (unsigned char ID, int MaxTorque)
int setSRL (unsigned char ID, unsigned char SRL)
int setRDT (unsigned char ID, unsigned char RDT)
int setLEDAlarm (unsigned char ID, unsigned char LEDAlarm)
int setShutdownAlarm (unsigned char ID, unsigned char SALARM)
int setCMargin (unsigned char ID, unsigned char CWCMargin, unsigned char CCWCMargin)
int setCSlope (unsigned char ID, unsigned char CWCSlope, unsigned char CCWCSlope)
int setPunch (unsigned char ID, int Punch)
int moving (unsigned char ID)
int lockRegister (unsigned char ID)
int RWStatus (unsigned char ID)
int readTemperature (unsigned char ID)
int readVoltage (unsigned char ID)

```

```
int readPosition (unsigned char ID)  
int readSpeed (unsigned char ID)  
int readLoad (unsigned char ID)  
int torqueStatus (unsigned char ID, bool Status)  
int ledStatus (unsigned char ID, bool Status)
```

Class EndStop

- Defined in file_arduino_common_EndStop.h

Class Documentation

class EndStop

Capteur fin de course est une classe permettant d'utiliser les capteurs fins de courses (clic de souris/bouton poussoir) Pour utiliser cette classe le bouton doit être d'un côté relié à la masse et de l'autre à l'arduino.

class *EndStop*

Public Functions

bool **getState** ()
Permet de connaître l'état courant du bouton.

Return etat (1 si enfoncé, 0 sinon).

void **attach** (int *pin*)
Indique le pin de l'arduino utilisé.

void **detach** ()
Réciproque de attach.

Class FullSpeedServo

- Defined in file_arduino_common_FullSpeedServo.h

Inheritance Relationships

Base Type

- public PeriodicProcess (*Class PeriodicProcess*)

Class Documentation

class FullSpeedServo : public *PeriodicProcess*

Pilotage de Servomoteur particulier.

class *FullSpeedServo* Cette classe permet de contrôler un Servomoteur comme un moteur continu avec une butée. Cela veut dire faire tourner un servomoteur dans une direction pendant un certain temps.

Public Functions

void **SpeedWrite** (int *setpoint*, float *time*)

Fait tourner le servomoteur pendant le temps donné ou jusqu'à la position demandée.

Parameters

- *setpoint*: position de commande
- *time*: durée en secondes

void **write** (int *setpoint*)

Fait tourner le servomoteur jusqu'à la position demandée (vitesse normale)

Parameters

- *setpoint*: position de commande

void **attach** (int *pin*)

Définit le pin sur lequel est connecté le servomoteur.

Parameters

- *pin*: de l'arduino

void **detach** ()

Fonction réciproque de *attach()* (permet par exemple d'avoir un servomoteur en roue libre)

bool **attached** ()

Permet de savoir si le servo est attaché ou pas (respectivement maintient sa position / roue libre)

Return renvoie 1 si attached et 0 si detached

int **read** ()

renvoie la dernière position envoyée au servo (seulement si le servo est "attached")

Return position du Servo entier

Protected Functions

void **process** (float *timestep*)

Méthode à implémenter obligatoirement pour hériter de *PeriodicProcess*.

Process est la méthode qui s'exécutera toutes les *m_timestep*. Il doit donc définir l'action répétitive voulue dans la loop de l'Arduino.

Parameters

- *timestep*: Temps écoulé depuis le dernier appel en seconde.

Class ArmManager

- Defined in file_esp32_common_IK_ArmManager.h

Inheritance Relationships

Base Type

- public IK::Picker (*Class Picker*)

Class Documentation

```
class IK::ArmManager : public IK::Picker
```

Public Functions

```
ArmManager ()
```

```
void set_workspace (Workspace ws_front, Workspace ws_back)
```

```
void set_origin (Coords origin)
```

```
void set_arm_link (float l1, float l2, float l3, int elbow_or)
```

```
void set_initial_joint_pos (Joints joints)
```

```
float get_link1 () const
```

```
float get_link2 () const
```

```
float get_link3 () const
```

```
float get_elbow () const
```

```
Workspace get_workspace_front () const
```

```
Workspace get_workspace_back () const
```

```
Coords get_origin () const
```

```
Joints get_joints () const
```

```
void init ()
```

```
Workspace clip_Workspaceo_constraints (Workspace workspace)
```

```
Coords workspace_center (Workspace workspace)
```

```

MoveBatch go_to (Coords start_pos, Coords target_pos)

float estimated_time_of_arrival (Coords start_pos, Coords start_vel, Coords target_pos, Coords target_vel)

void load (int address)

void save (int address) const

```

Class Joint

- Defined in file_esp32_common_IK_Joint.h

Class Documentation

```
class IK::Joint
```

Public Functions

```

Joint (int id, float pos_min, float pos_max, float velocity_min, float velocity_max, float acc_min, float acc_max)

vector_t get_path (float initial_pos, float initial_vel, float final_pos, float final_vel, float tf_sync, float delta_t)

TrajectoryTime time_to_destination (float initial_pos, float initial_vel, float final_pos, float final_vel)

```

Class Matrix2

- Defined in file_esp32_common_IK_Matrix.h

Class Documentation

```
class IK::Matrix2
```

Public Functions

```

matrix_t createMatrix22 (float X1, float X2, float Y1, float Y2)

matrix_t createMatrix21 (float X1, float Y1)

matrix_t multMatrix22x12 (matrix_t mat22, matrix_t mat12)

float norm (matrix_t mat)

float det (matrix_t mat)

matrix_t solve (matrix_t mat22, matrix_t mat12)

void free (matrix_t m)

```

Class Matrix3

- Defined in file_esp32_common_IK_Matrix.h

Class Documentation

```
class IK::Matrix3
```

Public Functions

```
matrix_t createMatrix33 (float X1, float X2, float X3, float Y1, float Y2, float Y3, float Z1, float Z2,
                          float Z3)
matrix_t createMatrix31 (float X1, float Y1, float Z1)
matrix_t multMatrix33x13 (matrix_t mat33, matrix_t mat13)
float norm (matrix_t mat)
float det (matrix_t mat)
matrix_t solve (matrix_t mat33, matrix_t mat13)
void free (matrix_t m)
```

Class MotorWrapper

- Defined in file_esp32_common_IK_MotorWrapper.h

Inheritance Relationships

Base Type

- public PeriodicProcess (*Class PeriodicProcess*)

Class Documentation

```
class IK::MotorWrapper : public PeriodicProcess
```

Public Functions

```
MotorWrapper ()
void setID (int id)
void setOFFSET (float offset)
int getID () const
float getOFFSET () const
void init ()
void setGoalPos (float pos)
```

```
void setVelocityProfile (vector<float> vel)
```

```
bool arrived () const
```

```
void process (float timestep)
```

Méthode à implémenter obligatoirement pour hériter de *PeriodicProcess*.

Process est la méthode qui s'exécutera toutes les *m_timestep*. Il doit donc définir l'action répétitive voulue dans la loop de l'Arduino.

Parameters

- *timestep*: Temps écoulé depuis le dernier appel en seconde.

```
void load (int address)
```

```
void save (int address) const
```

```
void end ()
```

Class Picker

- Defined in file_esp32_common_IK_Picker.h

Inheritance Relationships

Derived Type

- public IK::ArmManager (*Class ArmManager*)

Class Documentation

```
class IK::Picker
```

Subclassed by *IK::ArmManager*

Public Functions

```
void init (float l1, float l2, float l3, Joints joints, Coords origin, int elbow_or)
```

```
void flip_elbow (int elbow)
```

```
Coords forward_kinematics (Joints joints)
```

```
Joints inverse_kinematics (Coords tool)
```

```
Coords get_tool (void) const
```

```
Joints get_joints (void) const
```

```
DetailedPos get_detailed_pos (void) const
```

```
Coords get_tool_vel (Joints joints_vel)
```

```
Joints get_joints_vel (Coords tool_vel)
```

```
matrix_t compute_jacobian (void)
```

```
path_t get_path (Coords start_pos, Coords start_vel, Coords target_pos, Coords target_vel, float delta_t)
```

float **synchronisation_time** (*Joints start_pos, Joints start_vel, Joints target_pos, Joints target_vel*)

Public Members

float **_flip_elbow**

Constraints **x_axis**

Constraints **y_axis**

Constraints **phi_axis**

Class Scara

- Defined in file_esp32_common_IK_Scara.h

Class Documentation

class IK::Scara

Public Functions

Scara (float *l1*, float *l2*, *Joints joints*, *Coords origin*)

Coords **forward_kinematics** (*Joints joints*)

Joints **inverse_kinematics** (*Coords tool*)

Coords **get_tool** (void)

Joints **get_joints** (void)

DetailedPos **get_detailed_pos** (void)

matrix_t **compute_jacobian** (void)

Coords **get_tool_vel** (*Joints joints_vel*)

Joints **get_joints_vel** (*Coords tool_vel*)

path_t **get_path** (*Coords start_pos*, *Coords start_vel*, *Coords target_pos*, *Coords target_vel*, float *delta_t*)

float **synchronisation_time** (*Joints start_pos, Joints start_vel, Joints target_pos, Joints target_vel*)

Class TrajectoryManager

- Defined in file_esp32_common_IK_TrajectoryManager.h

Inheritance Relationships

Base Type

- `public PeriodicProcess (Class PeriodicProcess)`

Class Documentation

```
class IK::TrajectoryManager: public PeriodicProcess
```

Public Functions

```
TrajectoryManager ()
void set_armManager (ArmManager &manager)
void set_Motors (MotorWrapper &motor1, MotorWrapper &motor2, MotorWrapper &motor3)
void set_timestep (float timestep)
float get_timestep () const
void init ()
void move_directly (Coords pos)
bool is_arrived () const
void load (int address)
void save (int address) const
```

Class MatrixMath

- Defined in file_esp32_common_MatrixMath_MatrixMath.h

Class Documentation

```
class MatrixMath
```

Public Functions

```
void Print (float *A, int m, int n, String label)
void Copy (float *A, int n, int m, float *B)
void Multiply (float *A, float *B, int m, int p, int n, float *C)
void Add (float *A, float *B, int m, int n, float *C)
void Subtract (float *A, float *B, int m, int n, float *C)
void Transpose (float *A, int m, int n, float *C)
void Scale (float *A, int m, int n, float k)
int Invert (float *A, int n)
```

Class MoveBatch

- Defined in file_esp32_common_IK_MoveBatch.h

Class Documentation

class MoveBatch

Public Functions

MoveBatch()

void **addMove** (uint8_t *id*, float *pos*)

void **addVelocityProfile** (uint8_t *id*, vector<float> *vel*, vector<float> *time*)

void **addDuration** (float *time*)

bool **is_active** () **const**

float **get_duration** () **const**

Public Members

MoveCommand_t **batch**[3]

Class Mutex

- Defined in file_esp32_common_thread_tools.h

Class Documentation

class Mutex

Public Functions

Mutex()

bool **acquire** (int *wait_time* = -1) **const**

bool **release** () **const**

Class NonCopyable

- Defined in file_arduino_common_NonCopyable.h

Inheritance Relationships

Derived Types

- `private` `Codewheel` (*Class `Codewheel`*)
- `private` `DCMotor` (*Class `DCMotor`*)

Class Documentation

class `NonCopyable`

Classe à hériter pour empêcher la copie de cette dernière.

Subclassé par *`Codewheel`*, *`DCMotor`*

Protected Functions

`NonCopyable()`

`NonCopyable()`

Class Odometry

- Défini dans `file_arduino_common_Odometry.h`

Inheritance Relationships

Base Type

- `public` `PeriodicProcess` (*Class `PeriodicProcess`*)

Class Documentation

class `Odometry` : **public** *`PeriodicProcess`*

Calcule la position en temps réel du robot.

Author Ulysse Darmet *`Odometry`* est un *`PeriodicProcess`* qui calcule la position du robot à partir des roues codeuses (*`AbstractCodewheel`*).

Public Functions

void **setPosition** (float *x*, float *y*, float *theta*)

Attribut une nouvelle position.

À partir des coordonnées passer en paramètre, attribut les nouvelles coordonnées à sa *`Position`*.

Parameters

- *x*: Nouvelle coordonnée en x.
- *y*: Nouvelle coordonnée en y.

- `theta`: Nouvelle angle.

void **setAxleTrack** (float *axleTrack*)

Defini une nouvelle entraxe pour les roues codeuses.

Change l'entraxe actuel par celui indiqué en paramètre.

Parameters

- `axleTrack`: Nouvelle entraxe en mm.

void **setSlippage** (float *slippage*)

Defini la nouvelle dérive orthogonal.

Change la dérive orthogonal par celle indiquée en paramètre.

Parameters

- `slippage`: Nouvelle dérive orthogonal sans unité et signé.

void **setCodewheels** (*AbstractCodewheel* &*leftCodewheel*, *AbstractCodewheel* &*rightCodewheel*)

Defini les roues codeuses de *Odometry*.

Paramètre les pointeurs sur les deux *AbstractCodewheel* à utiliser pour le calcul d'odométrie.

Parameters

- `leftCodewheel`: *AbstractCodewheel* de la roue codeuse gauche.
- `rightCodewheel`: *AbstractCodewheel* de la roue codeuse droite.

const *Position* &**getPosition** () const

Retourne la position.

Retourne sa struc *Position* avec les dernières positions calculés.

Return La structure *Position*.

float **getLinVel** () const

Retourne la vitesse linéaire.

Rend la dernière vitesse linéaire calculé.

Return Vitesse lineaire en mm/s.

float **getAngVel** () const

Retourne la vitesse angulaire.

Rend la dernière vitesse angulaire calculé.

Return Vitesse angulaire en rad/s.

float **getAxleTrack** () const

Retourne l'entraxe utilisée.

Return Entraxe en mm.

float **getSlippage** () const

Retourne la dérive utilisée.

Return Dérive sans unité.

void **load** (int *address*)

Charge les paramètres.

Charge les paramètres depuis la mémoire de l'arduino.

Parameters

- *address*: Adresse à utilisé pour charger les données.

void **save** (int *address*) **const**

Sauvegarde les paramètres.

Sauvegarde les paramètres actuellement utilisés.

Parameters

- *address*: Adresse à utilisé pour la sauvegarde.

void **setPosition** (float *x*, float *y*, float *theta*)

void **setAxleTrack** (float *axleTrack*)

void **setSlippage** (float *slippage*)

void **setCodewheels** (*AbstractCodewheel* &*leftCodewheel*, *AbstractCodewheel* &*rightCodewheel*)

const *Position* **getPosition** () **const**

float **getLinVel** () **const**

float **getAngVel** () **const**

float **getAxleTrack** () **const**

float **getSlippage** () **const**

void **load** (int *address*)

void **save** (int *address*) **const**

Protected Functions

void **process** (float *timestep*)

Calcule la nouvelle position et la nouvelle vitesse. A partir de ses *AbstractCodewheel*, détermine la nouvelle vitesse instantanée et la nouvelle position.

Parameters

- *timestep*: Temps depuis le dernier appel de cette méthode en secondes.

void **process** (float *timestep*)

Méthode à implémenter obligatoirement pour hériter de *PeriodicProcess*.

Process est la méthode qui s'exécutera toutes les *m_timestep*. Il doit donc définir l'action répétitive voulue dans la loop de l'Arduino.

Parameters

- *timestep*: Temps écoulé depuis le dernier appel en seconde.

Protected Attributes

Position **m_pos**

Structure de position de *Odometry*.

float **m_linVel**

Vitesse lineaire en mm/s.

float **m_angVel**

Vitesse angulaire en rad/s.

float **m_axleTrack**

Entraxe entre les deux roues codeuses.

float **m_slippage**

Constante de dérivation othogonal.

AbstractCodewheel ***m_leftCodewheel**

Pointeur de l'*AbstractCodewheel* gauche.

AbstractCodewheel ***m_rightCodewheel**

Pointeur de l'*AbstractCodewheel* droite.

Mutex **m_mutex**

Class PeriodicProcess

- Defined in file_arduino_common_PeriodicProcess.h

Inheritance Relationships

Derived Types

- public BrushlessMotor (*Class BrushlessMotor*)
- public DifferentialController (*Class DifferentialController*)
- public FullSpeedServo (*Class FullSpeedServo*)
- public IK::MotorWrapper (*Class MotorWrapper*)
- public IK::TrajectoryManager (*Class TrajectoryManager*)
- public Odometry (*Class Odometry*)
- public PositionController (*Class PositionController*)
- public VelocityControllerLogs (*Class VelocityControllerLogs*)

Class Documentation

class `PeriodicProcess`

Classe à implémenter pour gérer les appels dans la loop.

class *PeriodicProcess* est un outil permettant à l'Arduino de pouvoir appeler l'objet *PeriodicProcess* tout les X s. Cela permet de ne pas saturer le microcontrôleur pour des tâches qui ne nécessitent pas un très gros rafraichissement. En général, on l'utilise pour toutes les tâches dans la loop.

Par exemple : faire tourner un moteur pendant un certain temps puis l'arreter en autonomie sans delay()

Subclassed by *BrushlessMotor*, *DifferentialController*, *FullSpeedServo*, *IK::MotorWrapper*, *IK::TrajectoryManager*, *Odometry*, *PositionController*, *VelocityControllerLogs*

Public Functions

`~PeriodicProcess()`

Constructeur de *PeriodicProcess*.

Le constructeur est totalement vide.

void `enable()`

Active le *PeriodicProcess*.

Passe la variable m_enable à Vrai et execute onProcessEnabling.

void `disable()`

Désactive le *PeriodicProcess*.

Passe la variable m_enable à Faux et execute onProcessDisabling.

void `setTimestep(float timestep)`

Sélectionne une nouvelle valeur pour timestep.

Change le timestep par celui donné en paramètre.

Parameters

- timestep: Temps en secondes du taux de rafraichissement.

bool `update()`

Execute la méthode process.

Execute la méthode process si le temps passé est supérieur à timestep. Envoie à process le temps depuis le dernier appel.

Return Vrai si process a été lancé et Faux sinon.

bool `isEnabled() const`

Vérifie si le *PeriodicProcess* est activé.

Return La valeur de m_enabled.

float `getTimestep() const`

Retourne la valeur de m_timestep.

Return La valeur de m_timestep.

`~PeriodicProcess()`

```
void enable ()  
void disable ()  
void setTimestep (float timestep)  
bool update ()  
bool isEnabled () const  
float getTimestep () const
```

Protected Functions

```
void process (float timestep) = 0
```

Méthode à implémenter obligatoirement pour hériter de *PeriodicProcess*.

Process est la méthode qui s'exécutera toutes les `m_timestep`. Il doit donc définir l'action répétitive voulue dans la loop de l'Arduino.

Parameters

- `timestep`: Temps écoulé depuis le dernier appel en seconde.

```
void onProcessEnabling ()
```

Méthode exécutée à l'activation du *PeriodicProcess*.

Méthode à implémenter si votre class nécessite des actions à son activation.

```
void onProcessDisabling ()
```

Méthode exécutée à la désactivation du *PeriodicProcess*.

Méthode à implémenter si votre class nécessite des actions à sa désactivation.

```
void process (float timestep) = 0
```

```
void onProcessEnabling ()
```

```
void onProcessDisabling ()
```

Class PID

- Defined in `file_arduino_common_PID.h`

Class Documentation

class PID

Classe d'asservissement.

Author Ulysse Darmet *PID* est une classe d'asservissement composée des 3 types d'asservissements. Elle permet à partir de l'erreur et de la constante désiré de retourner une commande asservie.

Public Functions

PID ()

Constructeur de *PID* Constructeur de *PID* qui initialise toutes les valeurs à des valeurs neutres ($K_p=1, K_i=0, K_d=0$).

float **compute** (float *setpoint*, float *input*, float *timestep*)

Calcul l'asservissement.

A partir de l'erreur et du temps depuis le dernier appel et de la constante demandé, rend une consigne asservie.

Return float La valeur asservie.

Parameters

- *setpoint*: Constante désiré.
- *input*: Constante actuel.
- *timestep*: Temps depuis le dernier appel en secondes.

void **reset** ()

Réinitialise les accumulateurs.

void **setTunings** (float *Kp*, float *Ki*, float *Kd*)

Charge de nouvelles constantes d'asservissements.

Parameters

- K_p : Coefficient proportionnel.
- K_i : Coefficient intégrateur.
- K_d : Coefficient dérivateur.

void **setOutputLimits** (float *minOutput*, float *maxOutput*)

Charge les limites de sorties.

Parameters

- *minOutput*: Minimum de sortie (peux être négatif).
- *maxOutput*: Maximum de sortie (peux être).

float **getKp** () **const**

Retourne le coefficient proportionnel.

Return Coefficient proportionnel.

float **getKi** () **const**

Retourne le coefficient intégrateur.

Return Coefficient intégrateur.

float **getKd** () **const**

Retourne le coefficient dérivateur.

Return Coefficient dérivateur.

float **getMinOutput** () **const**
Retourne la sortie minimal.

Return Sortie minimal.

float **getMaxOutput** () **const**
Retourne la sortie maximal.

Return Sortie maximal.

void **load** (int *address*)
Charge les paramètres de la mémoire.

Parameters

- *address*: Adresse à utiliser.

void **save** (int *address*) **const**
Sauvegarde les paramètres dans la mémoire.

Parameters

- *address*: Adresse à utiliser.

PID ()
Constructeur de *PID* Constructeur de *PID* qui initialise toutes les valeurs à des valeurs neutres ($K_p=1, K_i=0, K_d=0$).

float **compute** (float *setpoint*, float *input*, float *timestep*)
Calcul l'asservissement.

A partir de l'erreur et du temps depuis le dernier appel et de la constante demandé, rend une consigne asservie.

Return float La valeur asservie.

Parameters

- *setpoint*: Constante désiré.
- *input*: Constante actuel.
- *timestep*: Temps depuis le dernier appel en secondes.

void **reset** ()
Réinitialise les accumulateurs.

void **setTunings** (float *Kp*, float *Ki*, float *Kd*)
Charge de nouvelles constantes d'asservissements.

Parameters

- *Kp*: Coefficient proportionnel.
- *Ki*: Coefficient intégrateur.
- *Kd*: Coefficient dérivateur.

void **setOutputLimits** (float *minOutput*, float *maxOutput*)
Charge les limites de sorties.

Parameters

- `minOutput`: Minimum de sortie (peux être négatif).
- `maxOutput`: Maximum de sortie (peux être).

float **getKp** () **const**

Retourne le coefficient proportionnel.

Return Coefficient proportionnel.

float **getKi** () **const**

Retourne le coefficient intégrateur.

Return Coefficient intégrateur.

float **getKd** () **const**

Retourne le coefficient dérivateur.

Return Coefficient dérivateur.

float **getMinOutput** () **const**

Retourne la sortie minimal.

Return Sortie minimal.

float **getMaxOutput** () **const**

Retourne la sortie maximal.

Return Sortie maximal.

void **load** (int *address*)

Charge les paramètres de la mémoire.

Parameters

- `address`: Adresse à utiliser.

void **save** (int *address*) **const**

Sauvegarde les paramètres dans la mémoire.

Parameters

- `address`: Adresse à utiliser.

Class PositionController

- Defined in `file_arduino_common_PositionController.h`

Inheritance Relationships

Base Type

- `public PeriodicProcess (Class PeriodicProcess)`

Class Documentation

class PositionController : public *PeriodicProcess*

Classe support des objets *AbstractMoveStrategy*.

PositionController est le support des *AbstractMoveStrategy*. C'est à dire qu'il permet de charger ou supprimer une stratégie. Quand *PositionController* exécute une stratégie de mouvement, il va l'exécuté tous les `time_steps` pour y obtenir de nouvelles vitesses à suivre. *PositionController* va également renseigner la position du robot à *AbstractMoveStrategy* chargée.

Return class *PositionController* : public *PeriodicProcess* { public:

Public Functions

PositionController ()

Constructeur de *PositionController* Initialise les variables de *PositionController* à des valeurs neutre.

void **setPosInput** (const *Position* &posInput)

Charge les nouvelles positions du robot. Charge les nouvelles positions du robot pour les donner à une potentiel *AbstractMoveStrategy* chargée.

Parameters

- posInput: Nouvelle objet *Position* représentant la position du robot.

void **setPosSetpoint** (const *Position* &posSetpoint)

Charge la position à atteindre.

Charge la position à atteindre avec une *AbstractMoveStrategy*. Cette variable est potentiellement utilisé par l'*AbstractMoveStrategy*.

Parameters

- posSetpoint: *Position* à atteindre

void **setThetaSetpoint** (float theta)

Charge l'angle à atteindre.

Change l'angle objectif de l'objet *Position*. Remarque : les coordonnées objectif reste les mêmes.

Parameters

- theta: Nouvelle angle objectif.

float **getLinVelSetpoint** () const

Retourne la vitesse linéaire à atteindre.

Cette méthode retourne la vitesse linéaire que l'*AbstractMoveStrategy* souhaite atteindre.

Return Vitesse à atteindre en mm/s.

float **getAngVelSetpoint** () const

Retourne la vitesse angulaire à atteindre.

Cette méthode retourne la vitesse angulaire que l'*AbstractMoveStrategy* souhaite atteindre.

Return Vitesse angulaire à atteindre en rad/s.

void **setVelTunings** (float *linVelKp*, float *angVelKp*)

Paramètre les coefficients linéaire.

Paramètre de nouvelles valeurs pour les coefficients linéaire de vitesse et vitesse de rotation. Ces coefficients seront appliqués lors du calcul de vitesses à atteindre.

Parameters

- *linVelKp*: Coefficient proportionnel de vitesse linéaire.
- *angVelKp*: Coefficient proportionnel de vitesse angulaire.

void **setVelLimits** (float *linVelMax*, float *angVelMax*)

Paramètre les vitesses max.

Paramètre des vitesses maximaux qui doit être appliqué à l'*AbstractMoveStrategy*.

Parameters

- *linVelMax*: Vitesse linéaire max.
- *angVelMax*: Vitesse angulaire max.

void **setPosThresholds** (float *linPosThreshold*, float *angPosThreshold*)

Paramètre les précisions en position.

Paramètre les valeurs de précision pour l'*AbstractMoveStrategy*. Si l'erreur de position est inférieure, l'*AbstractMoveStrategy* est arrêté.

Parameters

- *linPosThreshold*: Précision en coordonnées cartésiennes (en mm).
- *angPosThreshold*: Précision d'angle (en rad).

void **setMoveStrategy** (*AbstractMoveStrategy* &*moveStrategy*)

Charge une stratégie de mouvement.

Charge la stratégie de mouvement (*AbstractMoveStrategy*) passée en paramètre. Elle sera active après l'activation de *PositionController* (*PositionController::enable*).

Parameters

- *moveStrategy*: *AbstractMoveStrategy* à utiliser.

bool **getPositionReached** ()

Indique si la position est atteinte.

Permet de savoir si la position objectif est atteinte avec la précision souhaitée.

Return true *Position* atteinte.

Return false *Position* non atteinte.

float **getLinVelKp** () **const**

Retourne le coef proportionnel de vitesse linéaire.

Return Coefficient proportionnel (sans unité).

float **getAngVelKp** () **const**

Retourne le coef proportionnel de vitesse angulaire.

Return Coefficient proportionnel (sans unité).

float **getLinVelMax** () **const**

Retourne la vitesse max linéaire.

Return Vitesse max en mm/s.

float **getAngVelMax** () **const**

Retourne la vitesse max angulaire.

Return Vitesse angulaire max en rad/s.

float **getLinPosThreshold** () **const**

Retourne la précision cartésienne.

Return Précision cartésienne en mm.

float **getAngPosThreshold** () **const**

Retourne la précision angulaire.

Return Précision angulaire en rad.

void **load** (int *address*)

Charge les configs.

Charge les configurations de la mémoire de l'Arduino avec l'adresse indiqué en paramètre.

Parameters

- *address*: Adresse à utiliser.

void **save** (int *address*) **const**

Sauvegarde la configuration actuel.

Parameters

- *address*: Adresse à utiliser.

PositionController ()

void **setPosInput** (const *Position* &*posInput*)

void **setPosSetpoint** (const *Position* &*posSetpoint*)

void **setThetaSetpoint** (float *theta*)

float **getLinVelSetpoint** () **const**

float **getAngVelSetpoint** () **const**

```
void setVelTunings (float linVelKp, float angVelKp)
void setVelLimits (float linVelMax, float angVelMax)
void setPosThresholds (float linPosThreshold, float angPosThreshold)
void setMoveStrategy (AbstractMoveStrategy &moveStrategy)
bool getPositionReached ()
float getLinVelKp () const
float getAngVelKp () const
float getLinVelMax () const
float getAngVelMax () const
float getLinPosThreshold () const
float getAngPosThreshold () const
void load (int address)
void save (int address) const
```

Class PressureSensor

- Defined in file_esp32_common_PressureSensor_PressureSensor.h

Class Documentation

```
class PressureSensor
```

Public Functions

```
PressureSensor (int analogPin)
int getPressurekPa ()
bool currentlyAtmospherePressure ()
void change_pressure_threshold (float threshold)
float get_pressure_threshold ()
```

Class PurePursuit

- Defined in file_arduino_common_PurePursuit.h

Nested Relationships

Nested Types

- *Struct PurePursuit::Waypoint*

Inheritance Relationships

Base Type

- `public AbstractMoveStrategy` (*Class AbstractMoveStrategy*)

Class Documentation

class PurePursuit : **public** *AbstractMoveStrategy*

Trajectoire courbe le long d'une ligne brisée.

class *PurePursuit* est un *AbstractMoveStrategy*.

Public Types

enum Direction

Sens de déplacement pour le robot.

enum Direction

Values:

enumerator FORWARD = 1

Le robot avance en marche avant.

enumerator BACKWARD = -1

Le robot avance en marche arrière.

enumerator FORWARD = 1

Le robot avance en marche avant.

enumerator BACKWARD = -1

Le robot avance en marche arrière.

enum Direction

Values:

enumerator FORWARD = 1

Le robot avance en marche avant.

enumerator BACKWARD = -1

Le robot avance en marche arrière.

enumerator FORWARD = 1

Le robot avance en marche avant.

enumerator BACKWARD = -1

Le robot avance en marche arrière.

Public Functions

PurePursuit ()

void **setDirection** (*Direction direction*)
 Setter du sens de marche du robot.

Parameters

- *direction*: Sens à utiliser pour le déplacement du robot.

void **setFinalAngle** (float *finalAngle*)
 Setter de l'angle à atteindre en fin de trajectoire.

Parameters

- *finalAngle*: Angle à atteindre (entre l'axe du robot et l'axe des x)

bool **addWaypoint** (const *Waypoint &waypoint*)
 Ajout un point en fin de ligne brisée.

Return true Ajout réussi.

Return false L'ajout a rencontré un problème.

Parameters

- *waypoint*: Nouveau point à ajouter.

void **reset** ()
 Initialise le Purpursuit. Supprime les points de la ligne brisée et les paramètres temporaires.

void **setLookAhead** (float *lookAhead*)
 Setter du lookAhead.

Met à jour la distance entre le point intermediaire et le robot.

Parameters

- *lookAhead*: Distance en mm.

void **setLookAheadBis** (float *lookAheadBis*)
 Setter du lookAheadBis.

Met à jour la distance entre le point intermediaire et le robot en fin de trajectoire.

Parameters

- *lookAheadBis*: Distance en mm.

Direction **getDirection** () const
 Getter de la direction courant.

Return Direction Sens utilisé.

float **getFinalAngle** () const
 Getter de l'angle final.

Retourne l'angle final que le robot va atteindre en fin de trajectoire.

Return float Angle final en rad.

const *Waypoint* &getWaypoint (int *index*) **const**

Getter d'un point de passage.

Retourne le point de passage sous la forme d'un *Waypoint*. L'index permet d'identifier le point à retourner.

Return *Waypoint*& Point de passage demandé.

Parameters

- *index*: Numéro du point à retourner.

int getNumWaypoints () **const**

Getter du nombre de point de passage.

Return int Nombre de points de la ligne brisée.

float getLookAhead () **const**

Getter du lookahead.

Return float LookAhead en mm.

float getLookAheadBis () **const**

Getter du LookAhead de fin de trajectoire.

Return float LookHeadbis en mm.

void load (int *address*)

Charge les paramètres sauvegardés.

Parameters

- *address*: Adresse à utiliser.

void save (int *address*) **const**

Sauvegarde les paramètres actuels.

Parameters

- *address*: Adresse à utiliser.

PurePursuit ()

void setDirection (*Direction* *direction*)

void setFinalAngle (float *finalAngle*)

bool addWaypoint (const *Waypoint* &waypoint)

void reset ()

void setLookAhead (float *lookAhead*)

void setLookAheadBis (float *lookAheadBis*)

Direction getDirection () **const**

float getFinalAngle () **const**

const *Waypoint* &getWaypoint (int *index*) **const**

```
int getNumWaypoints () const
float getLookAhead () const
float getLookAheadBis () const
void load (int address)
void save (int address) const
```

Protected Functions

```
void computeVelSetpoints (float timestep)
```

Calcul les nouvelles vitesses désirer.

Méthode à implémenter pour réaliser une *AbstractMoveStrategy*. Cette méthode calcul à partir de la position du robot des vitesses à suivre pour le robot.

Parameters

- *timestep*: Temps depuis le dernier appel en secondes.

```
bool getPositionReached ()
```

Indique si la position désirée est atteinte.

Calcul la distance entre la position du robot et la position désirée selon le mode de calcul de l'*AbstractMoveStrategy*.

Return true Si la position est atteinte.

Return false Si la position n'est pas atteinte.

```
bool checkLookAheadGoal (const float x, const float y)
```

Calcul le point intermediaire sur la ligne brisée.

Calcul le point intermediaire sur la courbe et met à jour le segment courant.

Return true Un point a été trouvé.

Return false Aucun point n'a été trouvé.

Parameters

- *x*: Coordonnées x du robot (mm).
- *y*: Coordonnées y du robot (mm).

```
void checkProjectionGoal (const float x, const float y)
```

Calcule le point intermediaire sur la ligne brisée.

Calcule le point le plus près du robot sur la ligne brisée non parcouru.

Parameters

- *x*: Coordonnées x du robot (mm).
- *y*: Coordonnées y du robot (mm).

```
float getDistAfterGoal ()
```

Retourne la distance restante à parcourir.

Return float Distance en mm à parcourir.

void **computeVelSetpoints** (float *timestep*)
Calcul les nouvelles vitesses désirer.

Méthode à implémenter pour réaliser une *AbstractMoveStrategy*. Cette méthode calcul à partir de la position du robot des vitesses à suivre pour le robot.

Parameters

- *timestep*: Temps depuis le dernier appel en secondes.

bool **getPositionReached** ()
Indique si la position désirée est atteinte.

Calcul la distance entre la position du robot et la position désirée selon le mode de calcul de l'*AbstractMoveStrategy*.

Return true Si la position est atteinte.

Return false Si la position n'est pas atteinte.

bool **checkLookAheadGoal** (const float x, const float y)

void **checkProjectionGoal** (const float x, const float y)

float **getDistAfterGoal** ()

Protected Attributes

Waypoint **m_waypoints**[16]
Liste des points de la ligne brisée à suivre.

int **m_numWaypoints**
Nombre de points constituant la ligne brisée.

Direction **m_direction**
Sens du robot pendant la trajectoire.

float **m_finalAngle**
Angle à atteindre en fin de trajectoire.

int **m_goalIndex**
Index courant.

float **m_goalParam**
Position relative de la projection du robot sur le segment courant.

bool **m_goalReached**
Arrivé ou non.

float **m_lookAhead**
Distance entre le point intermediaire et le robot.

float **m_lookAheadBis**
Distance entre le point intermediaire et le robot en fin de trajectoire.

struct Waypoint
Structure d'un point de passage de Pursuit.

struct *Waypoint*

Public Functions

```
Waypoint ()  
Waypoint (float x, float y)  
Waypoint (const Position &pos)  
Waypoint ()  
Waypoint (float x, float y)  
Waypoint (const Position &pos)
```

Public Members

```
float x  
float y
```

Class Semaphore

- Defined in file_esp32_common_thread_tools.h

Class Documentation

```
class Semaphore
```

Public Functions

```
Semaphore (int init_val, int max_val)  
bool acquire (int wait_time = -1) const  
bool release () const
```

Class SerialTalks

- Defined in file_arduino_common_SerialTalks.h

Nested Relationships

Nested Types

- *Class SerialTalks::ostream*

Class Documentation

class **SerialTalks**

Object de communication serial avec un ordinateur.

class *SerialTalks*

Author Ulysse Darmet

Author François Gauthier-Clerc est un outil permettant à l'arduino de pouvoir répondre aux requêtes reçu depuis le serial. Il utilise donc le port serial (usb) pour envoyer ou recevoir des données avec l'ordinateur ou la raspberry La classe est capable de lancer des methodes sur demande de l'ordinateur ou de la raspberry.

Public Types

typedef void (***Instruction**) (*SerialTalks* &inst, *Deserializer* &input, *Serializer* &output)

Instruction est un pointeur de fonction dont la signature doit être de la forme : (*SerialTalks*& inst, *Deserializer*& input, *Serializer*& output).

typedef void (***Processing**) (*SerialTalks* &inst, *Deserializer* &input)

Processing est un pointeur de fonction dont la signature doit être de la forme : (*SerialTalks*& inst, *Deserializer*& input). Cette méthode sera appelée après que la raspberry traitera la requête de l'arduino.

typedef void (***Instruction**) (*SerialTalks* &inst, *Deserializer* &input, *Serializer* &output)

typedef void (***Processing**) (*SerialTalks* &inst, *Deserializer* &input)

Public Functions

void **begin** (Stream &*stream*)

Initialise le *SerialTalks* avec un Stream d'<arduino.h>.

Parameters

- *stream*: Flux à associer pour la communication de *SerialTalks*.

void **bind** (*byte opcode*, *Instruction instruction*)

Associe une Instruction à un OPCODE.

Parameters

- *opcode*: Code à associer à la fonction.
- *instruction*: Fonction à répertorier dans *SerialTalks*.

void **attach** (*byte opcode*, *Processing processing*)

Associe une fonction au retour de la requête de l'OPCODE.

Parameters

- *opcode*: Code à associer à la fonction.
- *instruction*: Fonction à répertorier dans *SerialTalks*.

bool **execinstruction** (*byte *inputBuffer*)

Lance la fonction à partir des octets reçus. La méthode lit l'OPCode et transmet à la bonne fonction l'objet *Deserializer* avec le reste les octets reçu non traités et un *Serializer* pour la réponse à transmettre.

Return Vrai si la fonction à renvoyé des informations.

Parameters

- `inputBuffer`: Liste des octets reçus pour cette requête.

bool **execute** ()

Lit les octets reçus et les traite quand ils forment une requête complète.

Return Vrai si une requête à renvoyé une information.

Serializer **getSerializer** ()

Récupère le *Serializer* pour le remplir avant l'appel de la méthode `SerialTalks::send`.

Return *Serializer* à remplir.

int **send** (*byte opcode*, *Serializer output*)

Lance la requête avec les données chargées dans le *Serializer* et l'OPCODE.

Return int Nombre d'octet envoyés.

Parameters

- `opcode`: Code à utiliser pour le requête vers la Raspberry.
- `output`: *Serializer* à utiliser pour récupérer les données.

bool **isConnected** () const

Indique si le stream de *SerialTalks* est bien connecté.

Return Vrai si le stream est connecté.

bool **waitUntilConnected** (float *timeout* = -1)

Méthode bloquante jusqu'à la connexion du Stream ou jusqu'au timeout.

Return Vrai si le Stream est connecté.

Parameters

- `timeout`: Timeout pour la méthode

bool **getUUID** (char **uuid*)

Ecrit sur le pointeur l'UUID enregistré dans l'EEPROM de l'Arduino.

Return Vrai si il existe bien un UUID.

Parameters

- `uuid`: Pointeur à utiliser.

void **setUUID** (const char **uuid*)

Enregistre l'UUID dans l'EEPROM de l'Arduino.

Parameters

- `uuid`: Pointeur de l'UUID à enregistrer.

void **begin** (Stream &*stream*)

```
void bind (byte opcode, Instruction instruction)  
void attach (byte opcode, Processing processing)  
bool execinstruction (byte *inputBuffer)  
bool execute ()  
Serializer getSerializer ()  
int send (byte opcode, Serializer output)  
bool isConnected () const  
bool waitUntilConnected (float timeout = -1)  
bool getUUID (char *uuid)  
void setUUID (const char *uuid)
```

Public Members

ostream **out**

Flux virtuel pour les STD:OUT.

ostream **err**

Flux virtuel pour les STD:ERR ou erreur.

Public Static Functions

```
void generateRandomUUID (char *uuid, int length)  
Génère un UUID.
```

Parameters

- *uuid*: Pointeur pour renvoyer l'UUID.
- *length*: Longueur en octet de l'UUID à générer.

```
void generateRandomUUID (char *uuid, int length)
```

Protected Types

```
enum [anonymous]
```

Values:

```
enumerator SERIALTALKS_WAITING_STATE
```

En attente de l'arrivée d'un octet.

```
enumerator SERIALTALKS_INSTRUCTION_STARTING_STATE
```

En attente du prochain octet de la requête correspondant à la taille de celle-ci.

```
enumerator SERIALTALKS_CRC_RECEIVING_STATE
```

En attente du hash d'intégrité.

```
enumerator SERIALTALKS_INSTRUCTION_RECEIVING_STATE
```

Réception des derniers octet de la requête.

enum [anonymous]

Différents états de réception.

Values:

enumerator SERIALTALKS_ORDER

Requête reçu de la raspberry.

enumerator SERIALTALKS_RETURN

Retour de requête.

enum [anonymous]

Values:

enumerator SERIALTALKS_WAITING_STATE

En attente de l'arrivée d'un octet.

enumerator SERIALTALKS_INSTRUCTION_STARTING_STATE

En attente du prochain octet de la requête correspondant à la taille de celle-ci.

enumerator SERIALTALKS_CRC_RECEIVING_STATE

En attente du hash d'intégrité.

enumerator SERIALTALKS_INSTRUCTION_RECEIVING_STATE

Réception des derniers octet de la requête.

enum [anonymous]

Values:

enumerator SERIALTALKS_ORDER

Requête reçu de la raspberry.

enumerator SERIALTALKS_RETURN

Retour de requête.

Protected Functions

int **sendback** (long *retcode*, const *byte* **buffer*, int *size*)

bool **receive** (*byte* **inputBuffer*)

Méthode interne pour traiter les retours de requêtes.

Return true

Return false

Parameters

- *inputBuffer*:

int **sendback** (long *retcode*, const *byte* **buffer*, int *size*)

bool **receive** (*byte* **inputBuffer*)

Protected Attributes

Stream ***m_stream**

Stream de communication utilisé par *SerialTalks*.

bool **m_connected**

Représente l'état de connection.

Instruction **m_instructions**[0x20]

Listes des instructions enregistrées avec un OPCode associé.

Processing **m_processings**[0x4]

byte **m_inputBuffer**[64]

Listes des instructions de retour enregistrées avec un OPCode associé. Buffer d'entrée d'informations.

byte **m_outputBuffer**[64]

Buffer de sortie d'informations.

enum *SerialTalks::[anonymous]* **m_state**

enum *SerialTalks::[anonymous]* **m_order**

Différents états de réception.

byte **m_bytesNumber**

Type de requête reçu.

Variable pour la réception de données qui correspond à la longueur de la requête en bytes (valeur donnée dans le deuxième byte d'une requête).

byte **m_bytesCounter**

Variable d'incrementation pour la réception de données.

long **m_lastTime**

Timeout pour la réception d'octets d'une même requête.

unsigned long **m_lastRetcode**

CRC16 **m_crc**

byte **m_crcBytesCounter**

uint16_t **received_crc_value**

byte **m_crc_tab**[2 + 1]

byte **m_crc_tmp**[64]

enum *SerialTalks::[anonymous]* **m_state**

enum *SerialTalks::[anonymous]* **m_order**

Mutex **m_mutex**

class **ostream**: public Print

Stream virtuel pour les erreurs et autre. est un outils pour permettre de mieux transmettre les erreurs rencontrées et les STD::OUT.

class ostream

Public Functions

`size_t write (uint8_t)`

Ecrit sur le serial l'octet indiqué.

Return Nombre d'octet transmit.

Parameters

- `c`: octet à passer dans le serial.

`size_t write (const uint8_t *buffer, size_t size)`

Ecrit sur le serial le buffer indiqué (liste d'octets).

Return Nombre d'octet transmit.

Parameters

- `buffer`: à passer.
- `size`: (taille) du buffer.

`template<typename T>`

`ostream &operator<< (const T &object)`

Surcharge de l'opérateur '<<'. Cette méthode permet de passer plus facilement les objets dans le serial avec conversion en octets automatique.

Parameters

- `object`: à passer dans le serial.

`size_t write (uint8_t)`

`size_t write (const uint8_t *buffer, size_t size)`

`template<typename T>`

`ostream &operator<< (const T &object)`

Protected Functions

`void begin (SerialTalks &parent, long retcode)`

Initialise le ostream. C'est à dire expliciter le pointeur du *SerialTalks* et le retcode à associer.

Parameters

- `parent`: *SerialTalks* à associer.
- `retcode`: Code d'identification à utiliser pour l'utilisation du serial.

`void begin (SerialTalks &parent, long retcode)`

Protected Attributes

SerialTalks *`m_parent`

SerialTalks parent

long `m_retcode`

RetCode à associer au flux virtuel

Friends

friend class SerialTalks

Class SerialTalks::ostream

- Defined in file_arduino_common_SerialTalks.h

Nested Relationships

This class is a nested type of *Class SerialTalks*.

Inheritance Relationships

Base Type

- `public Print`

Class Documentation

class *SerialTalks::ostream* : **public** Print

Stream virtuel pour les erreurs et autre. est un outils pour permettre de mieux transmettre les erreurs rencontrées et les `STD::OUT`.

class ostream

Public Functions

size_t **write** (uint8_t)

Ecrit sur le serial l'octet indiqué.

Return Nombre d'octet transmit.

Parameters

- `c`: octet à passer dans le serial.

size_t **write** (**const** uint8_t **buffer*, size_t *size*)

Ecrit sur le serial le buffer indiqué (liste d'octets).

Return Nombre d'octet transmit.

Parameters

- `buffer`: à passer.
- `size`: (taille) du buffer.

template<typename **T**>

ostream &**operator**<< (**const** *T* &*object*)

Surcharge de l'opérateur '<<'. Cette méthode permet de passer plus facilement les objets dans le serial avec conversion en octets automatique.

Parameters

- `object`: à passer dans le serial.

`size_t write (uint8_t)`

`size_t write (const uint8_t *buffer, size_t size)`

`template<typename T>`

`ostream &operator<< (const T &object)`

Protected Functions

`void begin (SerialTalks &parent, long retcode)`

Initialise le ostream. C'est à dire expliciter le pointeur du *SerialTalks* et le retcode à associer.

Parameters

- `parent`: *SerialTalks* à associer.
- `retcode`: Code d'identification à utiliser pour l'utilisation du serial.

`void begin (SerialTalks &parent, long retcode)`

Protected Attributes

SerialTalks *`m_parent`

SerialTalks parent

long `m_retcode`

RetCode à associer au flux virtuel

Friends

`friend class SerialTalks`

Class SerialTopics

- Defined in file_arduino_common_SerialTopics.h

Nested Relationships

Nested Types

- *Struct SerialTopics::subscription_t*

Class Documentation

class SerialTopics

Public Types

typedef void (***Subscription**) (*Serializer* &output)
Subscription function pointer.

typedef void (***Subscription**) (*Serializer* &output)
Subscription function pointer.

Public Functions

void **begin** (*SerialTalks* &talks)
begin topics with serialtalks instance This function bind manage instruction and configure all topics by default

Parameters

- talks: *SerialTalks* instance

void **bind** (*byte opcode*, *Subscription subscription*)
Call bind function to associate custom topic at desired opcode.

Parameters

- opcode:
- subscription:

bool **execute** ()
function called at each loop iteration. This function check context for each topic and execute it when timeout occur.

Return true

Return false

subscription_t ***getSubscriptions** ()

void **begin** (*SerialTalks* &talks)

void **bind** (*byte opcode*, *Subscription subscription*)

bool **execute** ()

subscription_t ***getSubscriptions** ()

struct subscription_t
Subscription context structure.

Public Members

Subscription **func**

long **timestep**

long **lasttime**

bool **enable**

Class ShiftRegister

- Defined in file_arduino_common_ShiftRegister.h

Class Documentation

class ShiftRegister

Public Functions

void **attach** (uint8_t *latchpin*, uint8_t *clockpin*, uint8_t *datapin*)

void **SetHigh** (int *pos*)

void **SetLow** (int *pos*)

void **write** (int *pos*, int *state*)

void **shift** ()

void **attach** (uint8_t *latchpin*, uint8_t *clockpin*, uint8_t *datapin*)

void **SetHigh** (int *pos*)

void **SetLow** (int *pos*)

void **write** (int *pos*, int *state*)

void **shift** ()

Public Members

uint8_t **m_LATCH**

uint8_t **m_CLOCK**

uint8_t **m_DATA**

uint8_t **m_register**

Class StepByStepMotor

- Defined in file_arduino_common_StepByStepMotor.h

Class Documentation

class StepByStepMotor

Public Functions

```
StepByStepMotor ()  
void attach (int step, int dir, int enable, int rst, int sleep)  
void begin ()  
void step ()  
void update ()  
void set_position (double position)  
void set_speed (unsigned long speed)  
float get_position ()  
float get_speed ()  
void enable ()  
void disable ()
```

Class TaskManager

- Defined in file_esp32_common_TaskManager_TaskManager.h

Class Documentation

class TaskManager

Public Functions

```
TaskManager ()  
bool create_task (TaskFunction_t TaskCode, void *const Parameters)  
void delete_task ()  
bool task_is_running ()
```


Class Thread

- Defined in file_esp32_common_thread_tools.h

Class Documentation

class Thread

Public Functions

```
Thread (void *(*_funct)) void*  
    , void *args  
void kill ()  
void join ()
```

Class TurnOnTheSpot

- Defined in file_arduino_common_TurnOnTheSpot.h

Inheritance Relationships

Base Type

- public AbstractMoveStrategy (*Class AbstractMoveStrategy*)

Class Documentation

```
class TurnOnTheSpot : public AbstractMoveStrategy  
    Rotation du robot sans translations.  
    Class TurOnTheSpot
```

Public Types

```
enum Direction  
    Values:  
    enumerator TRIG = 1  
    enumerator CLOCK = -1
```

Public Functions

TurnOnTheSpot ()

void **setDirection** (*Direction direction*)

Protected Functions

void **computeVelSetpoints** (float *timestep*)

Calcul les nouvelles vitesses désirer.

Méthode à implémenter pour réaliser une *AbstractMoveStrategy*. Cette méthode calcul à partir de la position du robot des vitesses à suivre pour le robot.

Parameters

- *timestep*: Temps depuis le dernier appel en secondes.

bool **getPositionReached** ()

Indique si la position désirée est atteinte.

Calcul la distance entre la position du robot et la position désirée selon le mode de calcul de l'*AbstractMoveStrategy*.

Return true Si la position est atteinte.

Return false Si la position n'est pas atteinte.

void **computeVelSetpoints** (float *timestep*)

Calcul les nouvelles vitesses désirer.

Méthode à implémenter pour réaliser une *AbstractMoveStrategy*. Cette méthode calcul à partir de la position du robot des vitesses à suivre pour le robot.

Parameters

- *timestep*: Temps depuis le dernier appel en secondes.

bool **getPositionReached** ()

Indique si la position désirée est atteinte.

Calcul la distance entre la position du robot et la position désirée selon le mode de calcul de l'*AbstractMoveStrategy*.

Return true Si la position est atteinte.

Return false Si la position n'est pas atteinte.

Class VacuumPump

- Defined in file_esp32_common_VacuumPump_VacuumPump.h

Class Documentation

class VacuumPump

Public Functions

VacuumPump (int *vacumPin*, int *sluicePin*)

void **startPump** ()

void **stopPump** ()

void **startSluice** ()

void **stopSluice** ()

Class VelocityController

- Defined in file_arduino_common_VelocityController.h

Inheritance Relationships

Base Type

- public `DifferentialController` (*Class DifferentialController*)

Class Documentation

class VelocityController : public *DifferentialController*

Objet de controle de la vitesse.

VelocityController est une version améliorée de *DifferentialController* car il rajoute la gestion de Spin et les accélérations capées.

Public Functions

VelocityController ()

Constructeur de *VelocityController*.

Construteur de *VelocityController* qui initialise ces vairables sur des valeurs neutres.

void **setMaxAngAcc** (float *maxAngAcc*)

Paramètre les accélérations max.

Parameters

- *maxLinAcc*: Accélération linéaire en mm/s².
- *maxAngAcc*: Accélération angulaire en rad/s².

void **setMaxLinAcc** (float *maxLinAcc*)

void **setMaxLinDec** (float *maxLinDec*)
Paramètre les décélération max.

Parameters

- *maxLinDec*: Décélération linéaire en mm/s^2 .
- *maxAngDec*: Décélération angulaire en rad/s^2 .

void **setMaxAngDec** (float *maxAngDec*)

void **setSpinShutdown** (bool *spinShutdown*)
Change l'état de l'arrêt d'urgence.

Parameters

- *spinShutdown*: Etat à appliquer à la variable *spinShutdown*.

float **getMaxLinAcc** () **const**
Retourne l'accélération max linéaire.

Return Accélération en mm/s^2 .

float **getMaxAngAcc** () **const**
Retourne l'accélération max angulaire.

Return Accélération en rad/s^2 .

float **getMaxLinDec** () **const**
Retourne la décélération max linéaire.

Return Décélération en mm/s^2 .

float **getMaxAngDec** () **const**
Retourne la décélération max angulaire.

Return Décélération en rad/s^2 .

float **getLinSpinGoal** () **const**

float **getAngSpinGoal** () **const**

bool **getSpinShutdown** () **const**
Retourne l'état de *spinShutdown*.

Return true Si le robot est bloqué par un obstacle.

Return false Si le robot n'est pas bloqué.

void **load** (int *address*)
Charge les paramètres.

Charge les derniers paramètres sauvegarder (les acc et dec) dans l'Arduino.

Parameters

- *address*: Adresse à utiliser.

void **save** (int *address*) **const**

Sauvegarde les paramètres.

Sauvegarde les paramètres actuellement chargés.

Parameters

- *address*: Adresse à utiliser.

VelocityController ()

Constructeur de *VelocityController*.

Constructeur de *VelocityController* qui initialise ces variables sur des valeurs neutres.

void **setMaxAngAcc** (float *maxAngAcc*)

Paramètre les accélérations max.

Parameters

- *maxLinAcc*: Accélération linéaire en mm/s^2 .
- *maxAngAcc*: Accélération angulaire en rad/s^2 .

void **setMaxLinAcc** (float *maxLinAcc*)

void **setMaxLinDec** (float *maxLinDec*)

Paramètre les décélération max.

Parameters

- *maxLinDec*: Décélération linéaire en mm/s^2 .
- *maxAngDec*: Décélération angulaire en rad/s^2 .

void **setMaxAngDec** (float *maxAngDec*)

void **setSpinShutdown** (bool *spinShutdown*)

Change l'état de l'arrêt d'urgence.

Parameters

- *spinShutdown*: Etat à appliquer à la variable *spinShutdown*.

float **getMaxLinAcc** () **const**

Retourne l'accélération max linéaire.

Return Accélération en mm/s^2 .

float **getMaxAngAcc** () **const**

Retourne l'accélération max angulaire.

Return Accélération en rad/s^2 .

float **getMaxLinDec** () **const**

Retourne la décélération max linéaire.

Return Décélération en mm/s^2 .

float **getMaxAngDec** () **const**

Retourne la décélération max angulaire.

Return Décélération en rad/s^2 .

float **getLinSpinGoal** () **const**

float **getAngSpinGoal** () **const**

bool **getSpinShutdown** () **const**

Retourne l'état de spinShutDown.

Return true Si le robot est bloqué par un obstacle.

Return false Si le robot n'est pas bloqué.

void **load** (int *address*)

Charge les paramètres.

Charge les derniers paramètres sauvegarder (les acc et dec) dans l'Arduino.

Parameters

- *address*: Adresse à utiliser.

void **save** (int *address*) **const**

Sauvegarde les paramètres.

Sauvegarde les paramètres actuellement chargés.

Parameters

- *address*: Adresse à utiliser.

Protected Functions

float **genRampSetpoint** (float *stepSetpoint*, float *input*, float *rampSetpoint*, float *maxAcc*, float *maxDec*, float *timestep*)

Calcul la vitesse à atteindre.

Calcul les nouvelles vitesse à atteindre pour respecter les contraintes d'accélération.

Return float Nouvelle vitesse intermédiaire.

Parameters

- *stepSetpoint*: Vitesse demandé.
- *input*: Vitesse actuel
- *rampSetpoint*: Ancienne vitesse intermédiaire calculée.
- *maxAcc*: Accélération max.
- *maxDec*: Accélération min.
- *timestep*: Temps depuis le dernier appel.

void **process** (float *timestep*)

Calcul l'asservissement.

Parameters

- `timestep`: temps depuis le dernier appel.

void **onProcessEnabling** ()

Initialisation de l'asservissement.

float **genRampSetpoint** (float *stepSetpoint*, float *input*, float *rampSetpoint*, float *maxAcc*, float *maxDec*, float *timestep*)

Calcul la vitesse à atteindre.

Calcul les nouvelles vitesse à atteindre pour respecter les contraintes d'accélération.

Return float Nouvelle vitesse intermédiaire.

Parameters

- `stepSetpoint`: Vitesse demandé.
- `input`: Vitesse actuel
- `rampSetpoint`: Ancienne vitesse intermédiaire calculée.
- `maxAcc`: Accélération max.
- `maxDec`: Accélération min.
- `timestep`: Temps depuis le dernier appel.

void **process** (float *timestep*)

Calcul l'asservissement.

Parameters

- `timestep`: temps depuis le dernier appel.

void **onProcessEnabling** ()

Initialisation de l'asservissement.

Protected Attributes

float **m_rampLinVelSetpoint**

Vitesse linéaire intermédiaire en mm/s.

float **m_rampAngVelSetpoint**

Vitesse angulaire intermédiaire en rad/s.

float **m_maxLinAcc**

Accélération max linéaire en mm/s². Toujours positif.

float **m_maxLinDec**

Accélération max angulaire en rad/s². Toujours positif.

float **m_maxAngAcc**

Décélération max linéaire en mm/s². Toujours positif.

float **m_maxAngDec**

Décélération max angulaire en rad/s². Toujours positif.

bool **m_spinShutdown**

Etat de la sécurité de patinage.

float **m_linSpinGoal**

float **m_angSpinGoal**

Friends

friend class VelocityControllerLogs

Class VelocityControllerLogs

- Defined in file_esp32_common_VelocityController_VelocityController.h

Inheritance Relationships

Base Type

- public `PeriodicProcess` (*Class PeriodicProcess*)

Class Documentation

class `VelocityControllerLogs` : public *PeriodicProcess*

Classe d'enregistrement de vitesse.

Public Functions

void **setController** (**const** *VelocityController* &controller)

Protected Functions

void **process** (float *timestep*)

Méthode à implémenter obligatoirement pour hériter de *PeriodicProcess*.

Process est la méthode qui s'exécutera toutes les `m_timestep`. Il doit donc définir l'action répétitive voulue dans la loop de l'Arduino.

Parameters

- `timestep`: Temps écoulé depuis le dernier appel en seconde.

Protected Attributes

const *VelocityController* ***m_controller**

Mutex **m_mutex**

1.3.3 Enums

Enum `ax_error_t`

- Defined in `file_esp32_common_AX12_Dynamixel.h`

Enum Documentation

`enum ax_error_t`

Values:

```
enumerator INPUT_VOLTAGE = 1
enumerator ANGLE_LIMIT = 2
enumerator OVERHEATING = 4
enumerator RANGE = 8
enumerator CHECKSUM = 16
enumerator OVERLOAD = 32
enumerator INSTRUCTION = 64
```

1.3.4 Functions

Function `float_equals`

- Defined in `file_esp32_common_IK_ArmManager.cpp`

Function Documentation

`bool IK::float_equals (float a, float b, float epsilon = 0.001)`

Function `IK::equals`

- Defined in `file_esp32_common_IK_Picker.cpp`

Function Documentation

`bool IK::equals (float a, float b, float epsilon = EPSILON)`

Function IK::float_equals

- Defined in file_esp32_common_IK_MotorWrapper.cpp

Function Documentation

bool IK::float_equals (float *a*, float *b*, float *epsilon* = 0.001)

Template Function IK::operator<<(ostream&, const vector<T>&)

- Defined in file_esp32_common_IK_Joint.cpp

Function Documentation

Warning: doxygenfunction: Unable to resolve multiple matches for function “IK::operator<<” with arguments (ostream&, const vector<T>&) in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml. Potential matches:

```
- ostream &operator<<(ostream &out, const Coords &c)
- ostream &operator<<(ostream &out, const DetailedPos &d)
- ostream &operator<<(ostream &out, const Joints &j)
- ostream &operator<<(ostream &out, const TrajectoryTime &t)
- ostream &operator<<(ostream &out, const path_t &p)
- ostream &operator<<(ostream &out, const vector_t &v)
- template<typename T> ostream &operator<<(ostream &out, const vector<T> &v)
```

Function IK::operator<<(ostream&, const vector_t&)

- Defined in file_esp32_common_IK_Joint.cpp

Function Documentation

Warning: doxygenfunction: Unable to resolve multiple matches for function “IK::operator<<” with arguments (ostream&, const vector_t&) in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml. Potential matches:

```
- ostream &operator<<(ostream &out, const Coords &c)
- ostream &operator<<(ostream &out, const DetailedPos &d)
- ostream &operator<<(ostream &out, const Joints &j)
- ostream &operator<<(ostream &out, const TrajectoryTime &t)
- ostream &operator<<(ostream &out, const path_t &p)
- ostream &operator<<(ostream &out, const vector_t &v)
- template<typename T> ostream &operator<<(ostream &out, const vector<T> &v)
```

Function IK::operator<<(ostream&, const TrajectoryTime&)

- Defined in file_esp32_common_IK_Joint.cpp

Function Documentation

Warning: doxygenfunction: Unable to resolve multiple matches for function “IK::operator<<” with arguments (ostream&, const TrajectoryTime&) in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml. Potential matches:

```
- ostream &operator<<(ostream &out, const Coords &c)
- ostream &operator<<(ostream &out, const DetailedPos &d)
- ostream &operator<<(ostream &out, const Joints &j)
- ostream &operator<<(ostream &out, const TrajectoryTime &t)
- ostream &operator<<(ostream &out, const path_t &p)
- ostream &operator<<(ostream &out, const vector_t &v)
- template<typename T> ostream &operator<<(ostream &out, const vector<T> &v)
```

Function IK::operator<<(ostream&, const Coords&)

- Defined in file_esp32_common_IK_Picker.cpp

Function Documentation

Warning: doxygenfunction: Unable to resolve multiple matches for function “IK::operator<<” with arguments (ostream&, const Coords&) in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml. Potential matches:

```
- ostream &operator<<(ostream &out, const Coords &c)
- ostream &operator<<(ostream &out, const DetailedPos &d)
- ostream &operator<<(ostream &out, const Joints &j)
- ostream &operator<<(ostream &out, const TrajectoryTime &t)
- ostream &operator<<(ostream &out, const path_t &p)
- ostream &operator<<(ostream &out, const vector_t &v)
- template<typename T> ostream &operator<<(ostream &out, const vector<T> &v)
```

Function IK::operator<<(ostream&, const Joints&)

- Defined in file_esp32_common_IK_Picker.cpp

Function Documentation

Warning: doxygenfunction: Unable to resolve multiple matches for function “IK::operator<<” with arguments (ostream&, const Joints&) in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml. Potential matches:

```
- ostream &operator<<(ostream &out, const Coords &c)
- ostream &operator<<(ostream &out, const DetailedPos &d)
- ostream &operator<<(ostream &out, const Joints &j)
- ostream &operator<<(ostream &out, const TrajectoryTime &t)
- ostream &operator<<(ostream &out, const path_t &p)
- ostream &operator<<(ostream &out, const vector_t &v)
- template<typename T> ostream &operator<<(ostream &out, const vector<T> &v)
```

Function IK::operator<<(ostream&, const DetailedPos&)

- Defined in file_esp32_common_IK_Picker.cpp

Function Documentation

Warning: doxygenfunction: Unable to resolve multiple matches for function “IK::operator<<” with arguments (ostream&, const DetailedPos&) in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml. Potential matches:

```
- ostream &operator<<(ostream &out, const Coords &c)
- ostream &operator<<(ostream &out, const DetailedPos &d)
- ostream &operator<<(ostream &out, const Joints &j)
- ostream &operator<<(ostream &out, const TrajectoryTime &t)
- ostream &operator<<(ostream &out, const path_t &p)
- ostream &operator<<(ostream &out, const vector_t &v)
- template<typename T> ostream &operator<<(ostream &out, const vector<T> &v)
```

Function IK::operator<<(ostream&, const path_t&)

- Defined in file_esp32_common_IK_Picker.cpp

Function Documentation

Warning: doxygenfunction: Unable to resolve multiple matches for function “IK::operator<<” with arguments (ostream&, const path_t&) in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml. Potential matches:

```
- ostream &operator<<(ostream &out, const Coords &c)
- ostream &operator<<(ostream &out, const DetailedPos &d)
- ostream &operator<<(ostream &out, const Joints &j)
- ostream &operator<<(ostream &out, const TrajectoryTime &t)
- ostream &operator<<(ostream &out, const path_t &p)
```

```
- ostream &operator<<(ostream &out, const vector_t &v)
- template<typename T> ostream &operator<<(ostream &out, const vector<T> &v)
```

Function `inrange(float, float, float)`

- Defined in `file_arduino_common_mathutils.cpp`

Function Documentation

float **inrange** (float *x*, float *min*, float *max*)

Projet la variable *x* dans l'intervale *min*, *max*.

Cette fonction est très utile pour projeter des angles sur $-\pi / \pi$ par exemple.

Return float Valeur bornée sur l'intervalle.

Parameters

- *x*: Variable à projeter sur un intervalle.
- *min*: Valeur basse de l'intervalle.
- *max*: Valeur haute de l'intervalle.

Function `inrange(float, float, float)`

- Defined in `file_esp32_common_mathutils_mathutils.cpp`

Function Documentation

float **inrange** (float *x*, float *min*, float *max*)

Projet la variable *x* dans l'intervale *min*, *max*.

Cette fonction est très utile pour projeter des angles sur $-\pi / \pi$ par exemple.

Return float Valeur bornée sur l'intervalle.

Parameters

- *x*: Variable à projeter sur un intervalle.
- *min*: Valeur basse de l'intervalle.
- *max*: Valeur haute de l'intervalle.

Function `inrange(float, float, float)`

- Defined in `file_esp32_common_mathutils_mathutils.h`

Function Documentation

float **inrange** (float *x*, float *min*, float *max*)

Projet la variable *x* dans l'intervale *min*, *max*.

Cette fonction est très utile pour projeter des angles sur $-\pi / \pi$ par exemple.

Return float Valeur bornée sur l'intervalle.

Parameters

- *x*: Variable à projeter sur un intervalle.
- *min*: Valeur basse de l'intervalle.
- *max*: Valeur haute de l'intervalle.

Function `periodicmod(float, float)`

- Defined in `file_arduino_common_mathutils.cpp`

Function Documentation

float **periodicmod** (float *x*, float *y*)

Applique un modulo.

Return float

Parameters

- *x*:
- *y*:

Function `periodicmod(float, float)`

- Defined in `file_esp32_common_mathutils_mathutils.cpp`

Function Documentation

float **periodicmod** (float *x*, float *y*)

Applique un modulo.

Return float

Parameters

- *x*:
- *y*:

Function `periodicmod(float, float)`

- Defined in `file_esp32_common_mathutils_mathutils.h`

Function Documentation

float **periodicmod** (float *x*, float *y*)

Applique un modulo.

Return float

Parameters

- *x*:
- *y*:

Function `saturate(float, float, float)`

- Defined in `file_arduino_common_mathutils.cpp`

Function Documentation

float **saturate** (float *x*, float *min*, float *max*)

Borne la variable *x* entre les bornes.

Return float Valeur plaquée sur l'intervalle.

Parameters

- *x*: Variable à plaquer sur un intervalle.
- *min*: Valeur basse de l'intervalle.
- *max*: Valeur haute de l'intervalle.

Function `saturate(float, float, float)`

- Defined in `file_esp32_common_mathutils_mathutils.cpp`

Function Documentation

float **saturate** (float *x*, float *min*, float *max*)

Borne la variable *x* entre les bornes.

Return float Valeur plaquée sur l'intervalle.

Parameters

- *x*: Variable à plaquer sur un intervalle.
- *min*: Valeur basse de l'intervalle.
- *max*: Valeur haute de l'intervalle.

Function saturate(float, float, float)

- Defined in file_esp32_common_mathutils_mathutils.h

Function Documentation

float **saturate** (float *x*, float *min*, float *max*)

Borne la variable *x* entre les bornes.

Return float Valeur plaquée sur l'intervalle.

Parameters

- *x*: Variable à plaquer sur un intervalle.
- *min*: Valeur basse de l'intervalle.
- *max*: Valeur haute de l'intervalle.

Function shiftInSlow

- Defined in file_esp32_common_Codewheel_Codewheel.cpp

Function Documentation

Warning: doxygenfunction: Cannot find function “shiftInSlow” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Function sign(float)

- Defined in file_arduino_common_mathutils.cpp

Function Documentation

float **sign** (float *x*)

Retourne l'information du signe.

Return float 1 pour une variable positif -1 pour une négatif et 0 pour une variable nul (=0).

Parameters

- *x*: Variable à extraire le signe

Function sign(float)

- Defined in file_esp32_common_mathutils_mathutils.cpp

Function Documentation

float **sign** (float *x*)

Retourne l'information du signe.

Return float 1 pour une variable positif -1 pour une négatif et 0 pour une variable nul (=0).

Parameters

- *x*: Variable à extraire le signe

Function sign(float)

- Defined in file_esp32_common_mathutils_mathutils.h

Function Documentation

float **sign** (float *x*)

Retourne l'information du signe.

Return float 1 pour une variable positif -1 pour une négatif et 0 pour une variable nul (=0).

Parameters

- *x*: Variable à extraire le signe

1.3.5 Variables

Variable Dynamixel

- Defined in file_arduino_common_AX12_Dynamixel.cpp

Variable Documentation

DynamixelClass **Dynamixel**

Variable Dynamixel

- Defined in file_esp32_common_AX12_Dynamixel.cpp

Variable Documentation

DynamixelClass **Dynamixel**

Variable Dynamixel

- Defined in file_arduino_common_AX12_Dynamixel.h

Variable Documentation

DynamixelClass **Dynamixel**

Variable Dynamixel

- Defined in file_esp32_common_AX12_Dynamixel.h

Variable Documentation

DynamixelClass **Dynamixel**

Variable Matrix

- Defined in file_esp32_common_MatrixMath_MatrixMath.cpp

Variable Documentation

MatrixMath **Matrix**

Variable Matrix

- Defined in file_esp32_common_MatrixMath_MatrixMath.h

Variable Documentation

MatrixMath **Matrix**

Variable NULL_VEL

- Defined in file_esp32_common_IK_ArmManager.cpp

Variable Documentation

Warning: doxygenvariable: Cannot find variable “NULL_VEL” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Variable shift

- Defined in file_arduino_common_StepByStepMotor.cpp

Variable Documentation

Warning: doxygenvariable: Cannot find variable “shift” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Variable SoftSerial

- Defined in file_arduino_common_AX12_Dynamixel.cpp

Variable Documentation

Warning: doxygenvariable: Cannot find variable “SoftSerial” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Variable talks

- Defined in file_arduino_common_SerialTalks.cpp

Variable Documentation

SerialTalks **talks**

Variable talks

- Defined in file_esp32_common_SerialTalks_SerialTalks.cpp

Variable Documentation

SerialTalks **talks**

Variable talks

- Defined in file_arduino_common_SerialTalks.h

Variable Documentation

SerialTalks **talks**

Variable talks

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Variable Documentation

SerialTopics **topics**

Variable topics

- Defined in file_arduino_common_SerialTopics.cpp

Variable Documentation

SerialTopics **topics**

Variable topics

- Defined in file_esp32_common_SerialTopics_SerialTopics.cpp

Variable Documentation

SerialTopics **topics**

Variable topics

- Defined in file_arduino_common_SerialTopics.h

Variable Documentation

SerialTopics **topics**

Variable topics

- Defined in file_esp32_common_SerialTopics_SerialTopics.h

Variable Documentation

SerialTopics **topics**

1.3.6 Defines

Define _BV

- Defined in file_arduino_common_StepByStepMotor.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “_BV” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define ACC

- Defined in file_arduino_common_StepByStepMotor.h

Define Documentation

ACC

Define ANGLE_LIMIT_ERROR

- Defined in file_arduino_common_AX12_AX12.h

Define Documentation

ANGLE_LIMIT_ERROR (*x*)

Define `ANGLE_LIMIT_ERROR`

- Defined in file_esp32_common_AX12_AX12.h

Define Documentation

`ANGLE_LIMIT_ERROR` (*x*)

Define `ANGLE_LIMIT_ERROR_MASK`

- Defined in file_arduino_common_AX12_AX12.h

Define Documentation

`ANGLE_LIMIT_ERROR_MASK`

Define `ANGLE_LIMIT_ERROR_MASK`

- Defined in file_esp32_common_AX12_AX12.h

Define Documentation

`ANGLE_LIMIT_ERROR_MASK`

Define `availableData`

- Defined in file_arduino_common_AX12_Dynamixel.cpp

Define Documentation

<p>Warning: doxygendefine: Cannot find define “availableData” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml</p>
--

Define `availableData`

- Defined in file_esp32_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “availableData” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define AX_12_MAX_SPEED_RAD

- Defined in file_esp32_common_IK_Picker.h

Define Documentation

AX_12_MAX_SPEED_RAD

Define AX_ACTION

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_ACTION

Define AX_ACTION

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_ACTION

Define AX_ACTION_CHECKSUM

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_ACTION_CHECKSUM

Define **AX_ACTION_CHECKSUM**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_ACTION_CHECKSUM

Define **AX_ACTION_LENGTH**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_ACTION_LENGTH

Define **AX_ACTION_LENGTH**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_ACTION_LENGTH

Define **AX_ALARM_LED**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_ALARM_LED

Define **AX_ALARM_LED**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_ALARM_LED

Define AX_ALARM_SHUTDOWN

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_ALARM_SHUTDOWN****Define AX_ALARM_SHUTDOWN**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_ALARM_SHUTDOWN****Define AX_BAUD_RATE**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_BAUD_RATE****Define AX_BAUD_RATE**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_BAUD_RATE****Define AX_BD_LENGTH**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_BD_LENGTH**

Define AX_BD_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_BD_LENGTH

Define AX_BYTE_READ

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_BYTE_READ

Define AX_BYTE_READ

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_BYTE_READ

Define AX_BYTE_READ_POS

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_BYTE_READ_POS

Define AX_BYTE_READ_POS

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_BYTE_READ_POS

Define AX_CCW_AL_H

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_CCW_AL_H****Define AX_CCW_AL_H**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_CCW_AL_H****Define AX_CCW_AL_L**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_CCW_AL_L****Define AX_CCW_AL_L**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_CCW_AL_L****Define AX_CCW_ANGLE_LIMIT_H**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_CCW_ANGLE_LIMIT_H**

Define `AX_CCW_ANGLE_LIMIT_H`

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

`AX_CCW_ANGLE_LIMIT_H`

Define `AX_CCW_ANGLE_LIMIT_L`

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

`AX_CCW_ANGLE_LIMIT_L`

Define `AX_CCW_ANGLE_LIMIT_L`

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

`AX_CCW_ANGLE_LIMIT_L`

Define `AX_CCW_COMPLIANCE_MARGIN`

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

`AX_CCW_COMPLIANCE_MARGIN`

Define `AX_CCW_COMPLIANCE_MARGIN`

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

`AX_CCW_COMPLIANCE_MARGIN`

Define AX_CCW_COMPLIANCE_SLOPE

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_CCW_COMPLIANCE_SLOPE****Define AX_CCW_COMPLIANCE_SLOPE**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_CCW_COMPLIANCE_SLOPE****Define AX_CCW_CW_LENGTH**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_CCW_CW_LENGTH****Define AX_CCW_CW_LENGTH**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_CCW_CW_LENGTH****Define AX_CM_LENGTH**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_CM_LENGTH**

Define AX_CM_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_CM_LENGTH

Define AX_CS_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_CS_LENGTH

Define AX_CS_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_CS_LENGTH

Define AX_CW_ANGLE_LIMIT_H

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_CW_ANGLE_LIMIT_H

Define AX_CW_ANGLE_LIMIT_H

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_CW_ANGLE_LIMIT_H

Define AX_CW_ANGLE_LIMIT_L

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_CW_ANGLE_LIMIT_L****Define AX_CW_ANGLE_LIMIT_L**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_CW_ANGLE_LIMIT_L****Define AX_CW_COMPLIANCE_MARGIN**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_CW_COMPLIANCE_MARGIN****Define AX_CW_COMPLIANCE_MARGIN**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_CW_COMPLIANCE_MARGIN****Define AX_CW_COMPLIANCE_SLOPE**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_CW_COMPLIANCE_SLOPE**

Define AX_CW_COMPLIANCE_SLOPE

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_CW_COMPLIANCE_SLOPE

Define AX_DOWN_CALIBRATION_H

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_DOWN_CALIBRATION_H

Define AX_DOWN_CALIBRATION_H

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_DOWN_CALIBRATION_H

Define AX_DOWN_CALIBRATION_L

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_DOWN_CALIBRATION_L

Define AX_DOWN_CALIBRATION_L

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_DOWN_CALIBRATION_L

Define AX_DOWN_LIMIT_VOLTAGE

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_DOWN_LIMIT_VOLTAGE****Define AX_DOWN_LIMIT_VOLTAGE**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_DOWN_LIMIT_VOLTAGE****Define AX_GOAL_LENGTH**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_GOAL_LENGTH****Define AX_GOAL_LENGTH**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_GOAL_LENGTH****Define AX_GOAL_POSITION_H**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_GOAL_POSITION_H**

Define AX_GOAL_POSITION_H

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_GOAL_POSITION_H

Define AX_GOAL_POSITION_L

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_GOAL_POSITION_L

Define AX_GOAL_POSITION_L

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_GOAL_POSITION_L

Define AX_GOAL_SP_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_GOAL_SP_LENGTH

Define AX_GOAL_SP_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_GOAL_SP_LENGTH

Define AX_GOAL_SPEED_H

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_GOAL_SPEED_H****Define AX_GOAL_SPEED_H**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_GOAL_SPEED_H****Define AX_GOAL_SPEED_L**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_GOAL_SPEED_L****Define AX_GOAL_SPEED_L**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_GOAL_SPEED_L****Define AX_ID**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_ID**

Define AX_ID

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_ID

Define AX_ID_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_ID_LENGTH

Define AX_ID_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_ID_LENGTH

Define AX_LED

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_LED

Define AX_LED

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_LED

Define AX_LED_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_LED_LENGTH****Define AX_LED_LENGTH**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_LED_LENGTH****Define AX_LEDALARM_LENGTH**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_LEDALARM_LENGTH****Define AX_LEDALARM_LENGTH**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_LEDALARM_LENGTH****Define AX_LIMIT_TEMPERATURE**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_LIMIT_TEMPERATURE**

Define **AX_LIMIT_TEMPERATURE**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_LIMIT_TEMPERATURE

Define **AX_LOCK**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_LOCK

Define **AX_LOCK**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_LOCK

Define **AX_LR_LENGTH**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_LR_LENGTH

Define **AX_LR_LENGTH**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_LR_LENGTH

Define AX_MAX_TORQUE_H

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_MAX_TORQUE_H****Define AX_MAX_TORQUE_H**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_MAX_TORQUE_H****Define AX_MAX_TORQUE_L**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_MAX_TORQUE_L****Define AX_MAX_TORQUE_L**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_MAX_TORQUE_L****Define AX_MODEL_NUMBER_H**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_MODEL_NUMBER_H**

Define AX_MODEL_NUMBER_H

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_MODEL_NUMBER_H

Define AX_MODEL_NUMBER_L

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_MODEL_NUMBER_L
EEPROM AREA ///.

Define AX_MODEL_NUMBER_L

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_MODEL_NUMBER_L
EEPROM AREA ///.

Define AX_MOVING

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_MOVING

Define AX_MOVING

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_MOVING

Define AX_MOVING_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_MOVING_LENGTH

Define AX_MOVING_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_MOVING_LENGTH

Define AX_MT_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_MT_LENGTH

Define AX_MT_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_MT_LENGTH

Define AX_OPERATING_MODE

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_OPERATING_MODE

Define AX_OPERATING_MODE

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_OPERATING_MODE

Define AX_PAUSE_TIME

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_PAUSE_TIME

Define AX_PAUSE_TIME

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_PAUSE_TIME

Define AX_PING

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_PING

Instruction Set ///.

Define AX_PING

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_PING

Instruction Set ///.

Define AX_POS_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_POS_LENGTH

Define AX_POS_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_POS_LENGTH

Define AX_PRESENT_LOAD_H

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_PRESENT_LOAD_H

Define AX_PRESENT_LOAD_H

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_PRESENT_LOAD_H

Define AX_PRESENT_LOAD_L

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_PRESENT_LOAD_L

Define AX_PRESENT_LOAD_L

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_PRESENT_LOAD_L

Define AX_PRESENT_POSITION_H

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_PRESENT_POSITION_H

Define AX_PRESENT_POSITION_H

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_PRESENT_POSITION_H

Define AX_PRESENT_POSITION_L

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_PRESENT_POSITION_L

Define AX_PRESENT_POSITION_L

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_PRESENT_POSITION_L

Define AX_PRESENT_SPEED_H

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_PRESENT_SPEED_H

Define AX_PRESENT_SPEED_H

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_PRESENT_SPEED_H

Define AX_PRESENT_SPEED_L

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_PRESENT_SPEED_L

Define AX_PRESENT_SPEED_L

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_PRESENT_SPEED_L

Define AX_PRESENT_TEMPERATURE

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_PRESENT_TEMPERATURE

Define AX_PRESENT_TEMPERATURE

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_PRESENT_TEMPERATURE

Define AX_PRESENT_VOLTAGE

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_PRESENT_VOLTAGE

Define AX_PRESENT_VOLTAGE

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_PRESENT_VOLTAGE

Define AX_PUNCH_H

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_PUNCH_H

Define AX_PUNCH_H

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_PUNCH_H

Define AX_PUNCH_L

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_PUNCH_L

Define AX_PUNCH_L

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_PUNCH_L

Define AX_PUNCH_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_PUNCH_LENGTH

Define AX_PUNCH_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_PUNCH_LENGTH

Define AX_RDT_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_RDT_LENGTH

Define AX_RDT_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_RDT_LENGTH

Define AX_READ_DATA

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_READ_DATA

Define AX_READ_DATA

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_READ_DATA

Define AX_REG_WRITE

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_REG_WRITE

Define AX_REG_WRITE

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_REG_WRITE

Define AX_REGISTERED_INSTRUCTION

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_REGISTERED_INSTRUCTION

Define AX_REGISTERED_INSTRUCTION

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_REGISTERED_INSTRUCTION

Define AX_RESET

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_RESET

Define AX_RESET

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_RESET

Define AX_RESET_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_RESET_LENGTH

Define AX_RESET_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_RESET_LENGTH

Define AX_RETURN_ALL

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_RETURN_ALL

Define AX_RETURN_ALL

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_RETURN_ALL

Define AX_RETURN_DELAY_TIME

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_RETURN_DELAY_TIME

Define AX_RETURN_DELAY_TIME

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_RETURN_DELAY_TIME

Define AX_RETURN_LEVEL

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_RETURN_LEVEL

Define AX_RETURN_LEVEL

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_RETURN_LEVEL

Define AX_RETURN_NONE

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_RETURN_NONE

Status Return Levels ///.

Define AX_RETURN_NONE

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_RETURN_NONE

Status Return Levels ///.

Define AX_RETURN_READ

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_RETURN_READ

Define AX_RETURN_READ

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_RETURN_READ

Define AX_RWS_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_RWS_LENGTH

Define AX_RWS_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_RWS_LENGTH

Define AX_SALARM_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_SALARM_LENGTH

Define AX_SALARM_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_SALARM_LENGTH

Define AX_SPEED_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_SPEED_LENGTH

Define AX_SPEED_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_SPEED_LENGTH

Define AX_SRL_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_SRL_LENGTH

Define AX_SRL_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_SRL_LENGTH

Define AX_START

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_START

Define AX_START

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_START

Define AX_SYNC_WRITE

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_SYNC_WRITE

Define AX_SYNC_WRITE

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_SYNC_WRITE

Define AX_SYSTEM_DATA2

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_SYSTEM_DATA2

Define AX_SYSTEM_DATA2

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_SYSTEM_DATA2

Define AX_TEM_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_TEM_LENGTH

Define AX_TEM_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_TEM_LENGTH

Define AX_TL_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_TL_LENGTH

Define AX_TL_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_TL_LENGTH

Define AX_TORQUE_ENABLE

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_TORQUE_ENABLE

RAM AREA ///.

Define AX_TORQUE_ENABLE

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_TORQUE_ENABLE

RAM AREA ///.

Define AX_TORQUE_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_TORQUE_LENGTH

Define AX_TORQUE_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_TORQUE_LENGTH****Define AX_TORQUE_LIMIT_H**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_TORQUE_LIMIT_H****Define AX_TORQUE_LIMIT_H**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_TORQUE_LIMIT_H****Define AX_TORQUE_LIMIT_L**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_TORQUE_LIMIT_L****Define AX_TORQUE_LIMIT_L**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_TORQUE_LIMIT_L**

Define AX_UP_CALIBRATION_H

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_UP_CALIBRATION_H

Define AX_UP_CALIBRATION_H

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_UP_CALIBRATION_H

Define AX_UP_CALIBRATION_L

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_UP_CALIBRATION_L

Define AX_UP_CALIBRATION_L

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_UP_CALIBRATION_L

Define AX_UP_LIMIT_VOLTAGE

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_UP_LIMIT_VOLTAGE

Define AX_UP_LIMIT_VOLTAGE

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_UP_LIMIT_VOLTAGE****Define AX_VERSION**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_VERSION****Define AX_VERSION**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_VERSION****Define AX_VL_LENGTH**

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**AX_VL_LENGTH****Define AX_VL_LENGTH**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**AX_VL_LENGTH**

Define AX_VOLT_LENGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_VOLT_LENGTH

Define AX_VOLT_LENGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_VOLT_LENGTH

Define AX_WRITE_DATA

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

AX_WRITE_DATA

Define AX_WRITE_DATA

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

AX_WRITE_DATA

Define BACKWARD

- Defined in file_arduino_common_DCMotor.cpp

Define Documentation

<p>Warning: doxygendefine: Cannot find define “BACKWARD” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml</p>

Define BACKWARD

- Defined in file_esp32_common_DCMotor_DCMotor.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “BACKWARD” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define BACKWARD

- Defined in file_arduino_common_StepByStepMotor.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “BACKWARD” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define beginCom

- Defined in file_arduino_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “beginCom” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define beginCom

- Defined in file_esp32_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “beginCom” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define BROADCAST_ID

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

BROADCAST_ID

Define BROADCAST_ID

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

BROADCAST_ID

Define CHECKSUM_ERROR

- Defined in file_arduino_common_AX12_AX12.h

Define Documentation

CHECKSUM_ERROR (*x*)

Define CHECKSUM_ERROR

- Defined in file_esp32_common_AX12_AX12.h

Define Documentation

CHECKSUM_ERROR (*x*)

Define CHECKSUM_ERROR_MASK

- Defined in file_arduino_common_AX12_AX12.h

Define Documentation

CHECKSUM_ERROR_MASK

Define CHECKSUM_ERROR_MASK

- Defined in file_esp32_common_AX12_AX12.h

Define Documentation

CHECKSUM_ERROR_MASK

Define CRC_POLYNOME

- Defined in file_arduino_common_CRC16.h

Define Documentation

CRC_POLYNOME

Define CRC_POLYNOME

- Defined in file_esp32_common_CRC16_CRC16.h

Define Documentation

CRC_POLYNOME

Define DECC

- Defined in file_arduino_common_StepByStepMotor.h

Define Documentation

DECC

Define delayus

- Defined in file_arduino_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “delayus” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define delayus

- Defined in file_esp32_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “delayus” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define EEPROM_SIZE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

EEPROM_SIZE

Define ENABLE_VELOCITYCONTROLLER_LOGS

- Defined in file_arduino_common_VelocityController.h

Define Documentation

ENABLE_VELOCITYCONTROLLER_LOGS

Define ENABLE_VELOCITYCONTROLLER_LOGS

- Defined in file_esp32_common_VelocityController_VelocityController.h

Define Documentation

ENABLE_VELOCITYCONTROLLER_LOGS

Define endCom

- Defined in file_arduino_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “endCom” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define endCom

- Defined in file_esp32_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “endCom” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define FORWARD

- Defined in file_arduino_common_DCMotor.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “FORWARD” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define FORWARD

- Defined in file_esp32_common_DCMotor_DCMotor.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “FORWARD” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define FORWARD

- Defined in file_arduino_common_StepByStepMotor.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “FORWARD” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define INPUT_VOLTAGE_ERROR

- Defined in file_arduino_common_AX12_AX12.h

Define Documentation

INPUT_VOLTAGE_ERROR (*x*)

Define INPUT_VOLTAGE_ERROR

- Defined in file_esp32_common_AX12_AX12.h

Define Documentation

INPUT_VOLTAGE_ERROR (*x*)

Define INPUT_VOLTAGE_ERROR_MASK

- Defined in file_arduino_common_AX12_AX12.h

Define Documentation

INPUT_VOLTAGE_ERROR_MASK

Define INPUT_VOLTAGE_ERROR_MASK

- Defined in file_esp32_common_AX12_AX12.h

Define Documentation

INPUT_VOLTAGE_ERROR_MASK

Define INSTRUCTION_ERROR

- Defined in file_arduino_common_AX12_AX12.h

Define Documentation

INSTRUCTION_ERROR (*x*)

Define INSTRUCTION_ERROR

- Defined in file_esp32_common_AX12_AX12.h

Define Documentation

INSTRUCTION_ERROR (*x*)

Define INSTRUCTION_ERROR_MASK

- Defined in file_arduino_common_AX12_AX12.h

Define Documentation

INSTRUCTION_ERROR_MASK

Define INSTRUCTION_ERROR_MASK

- Defined in file_esp32_common_AX12_AX12.h

Define Documentation

INSTRUCTION_ERROR_MASK

Define LEFT

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

LEFT

Define LEFT

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

LEFT

Define LOCK

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

LOCK

Define LOCK

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

LOCK

Define LOG_ARM

- Defined in file_esp32_common_IK_ArmManager.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “LOG_ARM” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define LOG_JOINT

- Defined in file_esp32_common_IK_Joint.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “LOG_JOINT” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define LOG_MOTOR

- Defined in file_esp32_common_IK_MotorWrapper.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “LOG_MOTOR” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define LOG_PICKER

- Defined in file_esp32_common_IK_Picker.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “LOG_PICKER” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define LOG_TASK

- Defined in file_esp32_common_TaskManager_TaskManager.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “LOG_TASK” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define LOG_TRAJ

- Defined in file_esp32_common_IK_TrajectoryManager.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “LOG_TRAJ” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define MANAGE_OPCODE

- Defined in file_arduino_common_SerialTopics.h

Define Documentation

MANAGE_OPCODE

Define MANAGE_OPCODE

- Defined in file_esp32_common_SerialTopics_SerialTopics.h

Define Documentation

MANAGE_OPCODE

Define MAX

- Defined in file_esp32_common_AX12_AX12.cpp

Define Documentation

MAX (a, b)

Define MAX

- Defined in file_esp32_common_IK_MotorWrapper.h

Define Documentation

MAX (a, b)

Define MAX_JOINTS

- Defined in file_esp32_common_IK_MoveBatch.h

Define Documentation

MAX_JOINTS

Define MAX_NUM_OF_BATCHED_MOVES

- Defined in file_esp32_common_IK_TrajectoryManager.h

Define Documentation

MAX_NUM_OF_BATCHED_MOVES

Define MAX_PULSEWIDTH

- Defined in file_arduino_common_BrushlessMotor.h

Define Documentation

MAX_PULSEWIDTH

Define MAX_VELOCITY

- Defined in file_arduino_common_BrushlessMotor.h

Define Documentation

MAX_VELOCITY

Define MIN

- Defined in file_esp32_common_AX12_AX12.cpp

Define Documentation

MIN (*a*, *b*)

Define MIN

- Defined in file_esp32_common_IK_MotorWrapper.h

Define Documentation

MIN (*a*, *b*)

Define MIN_PULSEWIDTH

- Defined in file_arduino_common_BrushlessMotor.h

Define Documentation

MIN_PULSEWIDTH

Define MIN_VELOCITY

- Defined in file_arduino_common_BrushlessMotor.h

Define Documentation

MIN_VELOCITY

Define NR_END

- Defined in file_esp32_common_MatrixMath_MatrixMath.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “NR_END” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define OFF

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

OFF

Specials ///.

Define OFF

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

OFF

Specials ///.

Define ON

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

ON

Define ON

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

ON

Define OVERLOAD_ERROR

- Defined in file_arduino_common_AX12_AX12.h

Define Documentation

OVERLOAD_ERROR (*x*)

Define OVERLOAD_ERROR

- Defined in file_esp32_common_AX12_AX12.h

Define Documentation

OVERLOAD_ERROR (*x*)

Define OVERLOAD_ERROR_MASK

- Defined in file_arduino_common_AX12_AX12.h

Define Documentation

OVERLOAD_ERROR_MASK

Define OVERLOAD_ERROR_MASK

- Defined in file_esp32_common_AX12_AX12.h

Define Documentation

OVERLOAD_ERROR_MASK

Define P_MM

- Defined in file_arduino_common_StepByStepMotor.h

Define Documentation

P_MM

Define peekData

- Defined in file_arduino_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “peekData” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define peekData

- Defined in file_esp32_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “peekData” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define PLAT

- Defined in file_arduino_common_StepByStepMotor.h

Define Documentation

PLAT

Define PUREPURSUIT_MAX_WAYPOINTS

- Defined in file_arduino_common_PurePursuit.h

Define Documentation

PUREPURSUIT_MAX_WAYPOINTS

Define PUREPURSUIT_MAX_WAYPOINTS

- Defined in file_esp32_common_PurePursuit_PurePursuit.h

Define Documentation

PUREPURSUIT_MAX_WAYPOINTS

Define PWM_BIT

- Defined in file_esp32_common_DCMotor_DCMotor.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “PWM_BIT” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define RANGE_ERROR

- Defined in file_arduino_common_AX12_AX12.h

Define Documentation

RANGE_ERROR (*x*)

Define RANGE_ERROR

- Defined in file_esp32_common_AX12_AX12.h

Define Documentation

RANGE_ERROR (*x*)

Define RANGE_ERROR_MASK

- Defined in file_arduino_common_AX12_AX12.h

Define Documentation

RANGE_ERROR_MASK

Define RANGE_ERROR_MASK

- Defined in file_esp32_common_AX12_AX12.h

Define Documentation

RANGE_ERROR_MASK

Define readData

- Defined in file_arduino_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “readData” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml
--

Define readData

- Defined in file_esp32_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “readData” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml
--

Define RIGTH

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

RIGTH

Define RIGTH

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

RIGTH

Define RMP_TO_DEG_S

- Defined in file_esp32_common_IK_MotorWrapper.h

Define Documentation

RMP_TO_DEG_S

Define Rx_MODE

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

Rx_MODE

Define Rx_MODE

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

Rx_MODE

Define sendData

- Defined in file_arduino_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygengenerate: Cannot find define “sendData” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define sendData

- Defined in file_esp32_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygengenerate: Cannot find define “sendData” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define SERIALTALKS_BAUDRATE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_BAUDRATE

Baudrate utilisier

Define SERIALTALKS_BAUDRATE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_BAUDRATE

Baudrate utilisier

Define SERIALTALKS_CRC_SIZE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_CRC_SIZE

Define SERIALTALKS_CRC_SIZE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_CRC_SIZE

Define SERIALTALKS_DEFAULT_UUID_LENGTH

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_DEFAULT_UUID_LENGTH

Define SERIALTALKS_DEFAULT_UUID_LENGTH

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_DEFAULT_UUID_LENGTH

Define SERIALTALKS_DISCONNECT_OPCODE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_DISCONNECT_OPCODE

Define SERIALTALKS_DISCONNECT_OPCODE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_DISCONNECT_OPCODE

Define SERIALTALKS_FREE_BUFFER_OPCODE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_FREE_BUFFER_OPCODE

Define SERIALTALKS_FREE_BUFFER_OPCODE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_FREE_BUFFER_OPCODE

Define SERIALTALKS_GETBUFFERSIZE_OPCODE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_GETBUFFERSIZE_OPCODE

Define SERIALTALKS_GETBUFFERSIZE_OPCODE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_GETBUFFERSIZE_OPCODE

Define SERIALTALKS_GETEEPROM_OPCODE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_GETEEPROM_OPCODE

Define SERIALTALKS_GETEEPROM_OPCODE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_GETEEPROM_OPCODE

Define SERIALTALKS_GETUUID_OPCODE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_GETUUID_OPCODE

Define SERIALTALKS_GETUUID_OPCODE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_GETUUID_OPCODE

Define SERIALTALKS_INPUT_BUFFER_SIZE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_INPUT_BUFFER_SIZE

Define SERIALTALKS_INPUT_BUFFER_SIZE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_INPUT_BUFFER_SIZE

Define SERIALTALKS_MASTER_BYTE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_MASTER_BYTE

Define SERIALTALKS_MASTER_BYTE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_MASTER_BYTE

Define SERIALTALKS_MAX_OPCODE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_MAX_OPCODE

Define SERIALTALKS_MAX_OPCODE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_MAX_OPCODE

Define SERIALTALKS_MAX_PROCESSING

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_MAX_PROCESSING

Define SERIALTALKS_MAX_PROCESSING

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_MAX_PROCESSING

Define SERIALTALKS_OUTPUT_BUFFER_SIZE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_OUTPUT_BUFFER_SIZE

Define SERIALTALKS_OUTPUT_BUFFER_SIZE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_OUTPUT_BUFFER_SIZE

Define SERIALTALKS_PING_OPCODE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_PING_OPCODE

Define SERIALTALKS_PING_OPCODE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_PING_OPCODE

Define SERIALTALKS_RESEND_OPCODE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_RESEND_OPCODE

Define SERIALTALKS_RESEND_OPCODE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_RESEND_OPCODE

Define SERIALTALKS_RESEVED_OPCODE_0

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_0

Define SERIALTALKS_RESEVED_OPCODE_0

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_0

Define SERIALTALKS_RESEVED_OPCODE_1

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_1

Define SERIALTALKS_RESEVED_OPCODE_1

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_1

Define SERIALTALKS_RESEVED_OPCODE_2

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_2

Define SERIALTALKS_RESEVED_OPCODE_2

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_2

Define SERIALTALKS_RESEVED_OPCODE_3

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_3

Define SERIALTALKS_RESEVED_OPCODE_3

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_3

Define SERIALTALKS_RESEVED_OPCODE_4

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_4

Define SERIALTALKS_RESEVED_OPCODE_4

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_4

Define SERIALTALKS_RESEVED_OPCODE_5

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_5

Define SERIALTALKS_RESEVED_OPCODE_5

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_5

Define SERIALTALKS_RESEVED_OPCODE_6

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_6

Define SERIALTALKS_RESEVED_OPCODE_6

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_6

Define SERIALTALKS_RESEVED_OPCODE_7

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_7

Define SERIALTALKS_RESEVED_OPCODE_7

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_7

Define SERIALTALKS_RESEVED_OPCODE_8

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_8

Define SERIALTALKS_RESEVED_OPCODE_8

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_8

Define SERIALTALKS_RESEVED_OPCODE_9

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_9

Define SERIALTALKS_RESEVED_OPCODE_9

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_RESEVED_OPCODE_9

Define SERIALTALKS_SETEEPROM_OPCODE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_SETEEPROM_OPCODE

Define SERIALTALKS_SETEEPROM_OPCODE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_SETEEPROM_OPCODE

Define SERIALTALKS_SETUUID_OPCODE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_SETUUID_OPCODE

Define SERIALTALKS_SETUUID_OPCODE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_SETUUID_OPCODE

Define SERIALTALKS_SLAVE_BYTE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_SLAVE_BYTE

Define SERIALTALKS_SLAVE_BYTE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_SLAVE_BYTE

Define SERIALTALKS_STDERR_RETCODE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_STDERR_RETCODE

Define SERIALTALKS_STDERR_RETCODE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_STDERR_RETCODE

Define SERIALTALKS_STDOUT_RETCODE

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_STDOUT_RETCODE

Define SERIALTALKS_STDOUT_RETCODE

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_STDOUT_RETCODE

Define SERIALTALKS_UUID_ADDRESS

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_UUID_ADDRESS

Define SERIALTALKS_UUID_ADDRESS

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_UUID_ADDRESS

Define SERIALTALKS_UUID_LENGTH

- Defined in file_arduino_common_SerialTalks.h

Define Documentation

SERIALTALKS_UUID_LENGTH

Define SERIALTALKS_UUID_LENGTH

- Defined in file_esp32_common_SerialTalks_SerialTalks.h

Define Documentation

SERIALTALKS_UUID_LENGTH

Define SERIALTOPICS_DEFAULT_TIMING

- Defined in file_arduino_common_SerialTopics.h

Define Documentation

SERIALTOPICS_DEFAULT_TIMING

Define SERIALTOPICS_DEFAULT_TIMING

- Defined in file_esp32_common_SerialTopics_SerialTopics.h

Define Documentation

SERIALTOPICS_DEFAULT_TIMING

Define SERIALTOPICS_MAX_OPCODE

- Defined in file_arduino_common_SerialTopics.h

Define Documentation

SERIALTOPICS_MAX_OPCODE

Define SERIALTOPICS_MAX_OPCODE

- Defined in file_esp32_common_SerialTopics_SerialTopics.h

Define Documentation

SERIALTOPICS_MAX_OPCODE

Define setDPin

- Defined in file_arduino_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “setDPin” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define setDPin

- Defined in file_esp32_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “setDPin” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define setRXPin

- Defined in file_arduino_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “setRXPin” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define setTXPin

- Defined in file_arduino_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “setTXPin” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define STEP_BY_REV

- Defined in file_arduino_common_StepByStepMotor.h

Define Documentation

STEP_BY_REV

Define SUBSCRIBE

- Defined in file_arduino_common_SerialTopics.h

Define Documentation

SUBSCRIBE

Define SUBSCRIBE

- Defined in file_esp32_common_SerialTopics_SerialTopics.h

Define Documentation

SUBSCRIBE

Define switchCom

- Defined in file_arduino_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “switchCom” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define switchCom

- Defined in file_esp32_common_AX12_Dynamixel.cpp

Define Documentation

Warning: doxygendefine: Cannot find define “switchCom” in doxygen xml output for project “My Project” from directory: ./doxyoutput/xml

Define `TIME_OUT`

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

`TIME_OUT`

Define `TIME_OUT`

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

`TIME_OUT`

Define `TX_DELAY_TIME`

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation

`TX_DELAY_TIME`

Define `TX_DELAY_TIME`

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation

`TX_DELAY_TIME`

Define Tx_MODE

- Defined in file_arduino_common_AX12_Dynamixel.h

Define Documentation**Tx_MODE****Define Tx_MODE**

- Defined in file_esp32_common_AX12_Dynamixel.h

Define Documentation**Tx_MODE****Define UNSUBSCRIBE**

- Defined in file_arduino_common_SerialTopics.h

Define Documentation**UNSUBSCRIBE****Define UNSUBSCRIBE**

- Defined in file_esp32_common_SerialTopics_SerialTopics.h

Define Documentation**UNSUBSCRIBE****Define VELOCITYCONTROLLER_LOGS_Timestep**

- Defined in file_arduino_common_VelocityController.h

Define Documentation**VELOCITYCONTROLLER_LOGS_Timestep**

Define VELOCITYCONTROLLER_LOGS_Timestep

- Defined in file_esp32_common_VelocityController_VelocityController.h

Define Documentation

VELOCITYCONTROLLER_LOGS_Timestep

1.3.7 Typedefs

Typedef byte

- Defined in file_arduino_common_serialutils.h

Typedef Documentation

typedef unsigned char **byte**

Typedef byte

- Defined in file_esp32_common_serialutils.h

Typedef Documentation

typedef unsigned char **byte**

Typedef matrix_t

- Defined in file_esp32_common_IK_Matrix.h

Typedef Documentation

typedef float ****matrix_t**

Typedef String

- Defined in file_arduino_common_serialutils.h

Typedef Documentation

typedef std::string **String**

Typedef String

- Defined in file_esp32_common_serialutils.h

Typedef Documentation

typedef std::string **String**

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

A

- AbstractCodewheel (C++ class), 14
- AbstractCodewheel::~~AbstractCodewheel (C++ function), 14
- AbstractCodewheel::getTraveledDistance (C++ function), 14
- AbstractCodewheel::restart (C++ function), 14
- AbstractMotor (C++ class), 15
- AbstractMotor::~~AbstractMotor (C++ function), 15
- AbstractMotor::getMaxVelocity (C++ function), 15
- AbstractMotor::setVelocity (C++ function), 15
- AbstractMoveStrategy (C++ class), 16
- AbstractMoveStrategy::computeVelSetpoints (C++ function), 16, 17
- AbstractMoveStrategy::getAngPosThreshold (C++ function), 17
- AbstractMoveStrategy::getAngVelKp (C++ function), 16, 17
- AbstractMoveStrategy::getAngVelMax (C++ function), 17
- AbstractMoveStrategy::getLinPosThreshold (C++ function), 17
- AbstractMoveStrategy::getLinVelKp (C++ function), 16, 17
- AbstractMoveStrategy::getLinVelMax (C++ function), 17
- AbstractMoveStrategy::getPosInput (C++ function), 16, 17
- AbstractMoveStrategy::getPositionReached (C++ function), 16, 17
- AbstractMoveStrategy::getPosSetpoint (C++ function), 16, 17
- AbstractMoveStrategy::m_context (C++ member), 17
- AbstractMoveStrategy::setVelSetpoints (C++ function), 16, 17
- ACC (C macro), 97
- Acc (C++ struct), 2
- Acc::Acc (C++ function), 2
- Acc::max (C++ member), 2
- Acc::min (C++ member), 2
- ANGLE_LIMIT_ERROR (C macro), 97, 98
- ANGLE_LIMIT_ERROR_MASK (C macro), 98
- AX12 (C++ class), 18
- AX12::action (C++ function), 20
- AX12::attach (C++ function), 18, 19
- AX12::detach (C++ function), 18, 19
- AX12::end (C++ function), 20
- AX12::hold (C++ function), 19, 20
- AX12::isHolding (C++ function), 19, 20
- AX12::led (C++ function), 19, 20
- AX12::lockRegister (C++ function), 18, 19
- AX12::move (C++ function), 18, 19
- AX12::moveSpeed (C++ function), 18, 19
- AX12::moving (C++ function), 18, 19
- AX12::Nextmove (C++ function), 18, 19
- AX12::NextmoveSpeed (C++ function), 18, 19
- AX12::ping (C++ function), 18, 19
- AX12::readPosition (C++ function), 19, 20
- AX12::readSpeed (C++ function), 19, 20
- AX12::readTemperature (C++ function), 19
- AX12::readTorque (C++ function), 19, 20
- AX12::readVoltage (C++ function), 19, 20
- AX12::reset (C++ function), 19
- AX12::savedMove (C++ function), 18, 19
- AX12::SerialBegin (C++ function), 20
- AX12::setAngleLimit (C++ function), 18, 19
- AX12::setBD (C++ function), 18, 19
- AX12::setCMargin (C++ function), 18, 19
- AX12::setCSlope (C++ function), 18, 19
- AX12::setEndlessMode (C++ function), 18, 19
- AX12::setID (C++ function), 18, 19
- AX12::setLEDAlarm (C++ function), 18, 19
- AX12::setMaxTorque (C++ function), 18, 19
- AX12::setMaxTorqueRAM (C++ function), 18, 19
- AX12::setPunch (C++ function), 18, 19
- AX12::setRDT (C++ function), 18, 19
- AX12::setShutdownAlarm (C++ function), 18, 19
- AX12::setSRL (C++ function), 18, 19
- AX12::setTempLimit (C++ function), 18, 19

AX12::setVoltageLimit (C++ function), 18, 19
AX12::turn (C++ function), 18, 19
AX12error (C++ class), 20
AX12error::AX12error (C++ function), 20
AX12error::get_error_code (C++ function), 20
AX12error::get_id (C++ function), 20
AX12error::resolve_AX_error (C++ function), 20
AX12Timeout (C++ class), 21
AX12Timeout::AX12Timeout (C++ function), 21
AX12Timeout::get_id (C++ function), 21
AX_12_MAX_SPEED_RAD (C macro), 99
AX_ACTION (C macro), 99
AX_ACTION_CHECKSUM (C macro), 99, 100
AX_ACTION_LENGTH (C macro), 100
AX_ALARM_LED (C macro), 100
AX_ALARM_SHUTDOWN (C macro), 101
AX_BAUD_RATE (C macro), 101
AX_BD_LENGTH (C macro), 101, 102
AX_BYTE_READ (C macro), 102
AX_BYTE_READ_POS (C macro), 102
AX_CCW_AL_H (C macro), 103
AX_CCW_AL_L (C macro), 103
AX_CCW_ANGLE_LIMIT_H (C macro), 103, 104
AX_CCW_ANGLE_LIMIT_L (C macro), 104
AX_CCW_COMPLIANCE_MARGIN (C macro), 104
AX_CCW_COMPLIANCE_SLOPE (C macro), 105
AX_CCW_CW_LENGTH (C macro), 105
AX_CM_LENGTH (C macro), 105, 106
AX_CS_LENGTH (C macro), 106
AX_CW_ANGLE_LIMIT_H (C macro), 106
AX_CW_ANGLE_LIMIT_L (C macro), 107
AX_CW_COMPLIANCE_MARGIN (C macro), 107
AX_CW_COMPLIANCE_SLOPE (C macro), 107, 108
AX_DOWN_CALIBRATION_H (C macro), 108
AX_DOWN_CALIBRATION_L (C macro), 108
AX_DOWN_LIMIT_VOLTAGE (C macro), 109
ax_error_t (C++ enum), 85
ax_error_t::ANGLE_LIMIT (C++ enumerator), 85
ax_error_t::CHECKSUM (C++ enumerator), 85
ax_error_t::INPUT_VOLTAGE (C++ enumerator), 85
ax_error_t::INSTRUCTION (C++ enumerator), 85
ax_error_t::OVERHEATING (C++ enumerator), 85
ax_error_t::OVERLOAD (C++ enumerator), 85
ax_error_t::RANGE (C++ enumerator), 85
AX_GOAL_LENGTH (C macro), 109
AX_GOAL_POSITION_H (C macro), 109, 110
AX_GOAL_POSITION_L (C macro), 110
AX_GOAL_SP_LENGTH (C macro), 110
AX_GOAL_SPEED_H (C macro), 111
AX_GOAL_SPEED_L (C macro), 111
AX_ID (C macro), 111, 112
AX_ID_LENGTH (C macro), 112
AX_LED (C macro), 112
AX_LED_LENGTH (C macro), 113
AX_LEDALARM_LENGTH (C macro), 113
AX_LIMIT_TEMPERATURE (C macro), 113, 114
AX_LOCK (C macro), 114
AX_LR_LENGTH (C macro), 114
AX_MAX_TORQUE_H (C macro), 115
AX_MAX_TORQUE_L (C macro), 115
AX_MODEL_NUMBER_H (C macro), 115, 116
AX_MODEL_NUMBER_L (C macro), 116
AX_MOVING (C macro), 116, 117
AX_MOVING_LENGTH (C macro), 117
AX_MT_LENGTH (C macro), 117
AX_OPERATING_MODE (C macro), 118
AX_PAUSE_TIME (C macro), 118
AX_PING (C macro), 118, 119
AX_POS_LENGTH (C macro), 119
AX_PRESENT_LOAD_H (C macro), 119, 120
AX_PRESENT_LOAD_L (C macro), 120
AX_PRESENT_POSITION_H (C macro), 120
AX_PRESENT_POSITION_L (C macro), 121
AX_PRESENT_SPEED_H (C macro), 121
AX_PRESENT_SPEED_L (C macro), 121, 122
AX_PRESENT_TEMPERATURE (C macro), 122
AX_PRESENT_VOLTAGE (C macro), 122
AX_PUNCH_H (C macro), 123
AX_PUNCH_L (C macro), 123
AX_PUNCH_LENGTH (C macro), 123, 124
AX_RDT_LENGTH (C macro), 124
AX_READ_DATA (C macro), 124
AX_REG_WRITE (C macro), 125
AX_REGISTERED_INSTRUCTION (C macro), 125
AX_RESET (C macro), 125, 126
AX_RESET_LENGTH (C macro), 126
AX_RETURN_ALL (C macro), 126
AX_RETURN_DELAY_TIME (C macro), 127
AX_RETURN_LEVEL (C macro), 127
AX_RETURN_NONE (C macro), 127, 128
AX_RETURN_READ (C macro), 128
AX_RWS_LENGTH (C macro), 128, 129
AX_SALARM_LENGTH (C macro), 129
AX_SPEED_LENGTH (C macro), 129
AX_SRL_LENGTH (C macro), 130
AX_START (C macro), 130
AX_SYNC_WRITE (C macro), 130, 131
AX_SYSTEM_DATA2 (C macro), 131
AX_TEM_LENGTH (C macro), 131
AX_TL_LENGTH (C macro), 132
AX_TORQUE_ENABLE (C macro), 132
AX_TORQUE_LENGTH (C macro), 132, 133
AX_TORQUE_LIMIT_H (C macro), 133

AX_TORQUE_LIMIT_L (*C macro*), 133
 AX_UP_CALIBRATION_H (*C macro*), 134
 AX_UP_CALIBRATION_L (*C macro*), 134
 AX_UP_LIMIT_VOLTAGE (*C macro*), 134, 135
 AX_VERSION (*C macro*), 135
 AX_VL_LENGTH (*C macro*), 135
 AX_VOLT_LENGTH (*C macro*), 136
 AX_WRITE_DATA (*C macro*), 136

B

BROADCAST_ID (*C macro*), 138
 BrushlessMotor (*C++ class*), 21
 BrushlessMotor::attach (*C++ function*), 21
 BrushlessMotor::BrushlessMotor (*C++ function*), 21
 BrushlessMotor::detach (*C++ function*), 21
 BrushlessMotor::disableMotor (*C++ function*), 21
 BrushlessMotor::disableStartup (*C++ function*), 21
 BrushlessMotor::enableMotor (*C++ function*), 21
 BrushlessMotor::enableStartup (*C++ function*), 21
 BrushlessMotor::forcePulsewidth (*C++ function*), 21
 BrushlessMotor::getVelocity (*C++ function*), 21
 BrushlessMotor::isEnabled (*C++ function*), 22
 BrushlessMotor::process (*C++ function*), 22
 BrushlessMotor::readMicroseconds (*C++ function*), 22
 BrushlessMotor::setPulsewidth (*C++ function*), 21
 BrushlessMotor::setVelocity (*C++ function*), 21
 BrushlessMotor::startupProcess (*C++ function*), 21
 BrushlessMotor::update (*C++ function*), 21
 BrushlessMotor::updateStartup (*C++ function*), 21
 byte (*C++ type*), 172

C

CHECKSUM_ERROR (*C macro*), 138
 CHECKSUM_ERROR_MASK (*C macro*), 138, 139
 Clock (*C++ class*), 22
 Clock::Clock (*C++ function*), 22
 Clock::getElapsedTime (*C++ function*), 22
 Clock::restart (*C++ function*), 22
 Codewheel (*C++ class*), 23
 Codewheel::attachCounter (*C++ function*), 23, 25

Codewheel::attachRegister (*C++ function*), 23, 25
 Codewheel::Codewheel (*C++ function*), 23, 25
 Codewheel::getCounter (*C++ function*), 23, 25
 Codewheel::getCountsPerRev (*C++ function*), 24, 25
 Codewheel::getTraveledDistance (*C++ function*), 24, 26
 Codewheel::getWheelRadius (*C++ function*), 24, 25
 Codewheel::load (*C++ function*), 24, 26
 Codewheel::m_COUNTER_AXIS (*C++ member*), 26
 Codewheel::m_COUNTER_OE (*C++ member*), 27
 Codewheel::m_COUNTER_RST (*C++ member*), 27
 Codewheel::m_COUNTER_SEL1 (*C++ member*), 27
 Codewheel::m_COUNTER_SEL2 (*C++ member*), 27
 Codewheel::m_COUNTER_XY (*C++ member*), 26
 Codewheel::m_countsPerRev (*C++ member*), 26
 Codewheel::m_currentCounter (*C++ member*), 26
 Codewheel::m_REGISTER_CLOCK (*C++ member*), 27
 Codewheel::m_REGISTER_DATA (*C++ member*), 27
 Codewheel::m_REGISTER_LATCH (*C++ member*), 27
 Codewheel::m_startCounter (*C++ member*), 26
 Codewheel::m_wheelRadius (*C++ member*), 26
 Codewheel::reset (*C++ function*), 24, 26
 Codewheel::restart (*C++ function*), 24, 26
 Codewheel::save (*C++ function*), 24, 26
 Codewheel::setCountsPerRev (*C++ function*), 24, 25
 Codewheel::setWheelRadius (*C++ function*), 24, 25
 Codewheel::update (*C++ function*), 26
 Constraints (*C++ struct*), 3
 Constraints::acc (*C++ member*), 3
 Constraints::Constraints (*C++ function*), 3
 Constraints::pos (*C++ member*), 3
 Constraints::vel (*C++ member*), 3
 Coords (*C++ struct*), 3
 Coords::Coords (*C++ function*), 3
 Coords::phi (*C++ member*), 3
 Coords::x (*C++ member*), 3
 Coords::y (*C++ member*), 3
 CRC16 (*C++ class*), 27
 CRC16::CRC16 (*C++ function*), 27
 CRC16::CRCcheck (*C++ function*), 27
 CRC16::CRCprocessBuffer (*C++ function*), 27
 CRC16::CRCprocessByte (*C++ function*), 27
 CRC_POLYNOME (*C macro*), 139

D

DCMotor (*C++ class*), 28
DCMotor::attach (*C++ function*), 28, 29
DCMotor::DCMotor (*C++ function*), 28, 29
DCMotor::disable (*C++ function*), 29, 30
DCMotor::enable (*C++ function*), 29, 30
DCMotor::getConstant (*C++ function*), 29, 30
DCMotor::getMaxPWM (*C++ function*), 29, 30
DCMotor::getMaxVelocity (*C++ function*), 29, 30
DCMotor::getVelocity (*C++ function*), 29, 30
DCMotor::getWheelRadius (*C++ function*), 29, 30
DCMotor::isEnabled (*C++ function*), 29, 30
DCMotor::load (*C++ function*), 29, 30
DCMotor::m_constant (*C++ member*), 31
DCMotor::m_DIR (*C++ member*), 31
DCMotor::m_EN (*C++ member*), 31
DCMotor::m_enabled (*C++ member*), 31
DCMotor::m_maxPWM (*C++ member*), 31
DCMotor::m_mutex (*C++ member*), 31
DCMotor::m_PWM (*C++ member*), 31
DCMotor::m_PWMChanel (*C++ member*), 31
DCMotor::m_velocity (*C++ member*), 31
DCMotor::m_wheelRadius (*C++ member*), 31
DCMotor::save (*C++ function*), 29, 30
DCMotor::setConstant (*C++ function*), 28, 29
DCMotor::setMaxPWM (*C++ function*), 28, 30
DCMotor::setVelocity (*C++ function*), 28, 29
DCMotor::setWheelRadius (*C++ function*), 28, 29
DCMotor::update (*C++ function*), 30
DCMotorsDriver (*C++ class*), 31
DCMotorsDriver::attach (*C++ function*), 31
DCMotorsDriver::isFaulty (*C++ function*), 31, 32
DCMotorsDriver::reset (*C++ function*), 31, 32
DECC (*C macro*), 139
Deserializer (*C++ struct*), 4
Deserializer::adr (*C++ member*), 5
Deserializer::buffer (*C++ member*), 5
Deserializer::clear (*C++ function*), 4
Deserializer::Deserializer (*C++ function*), 4
Deserializer::operator>> (*C++ function*), 4
Deserializer::read (*C++ function*), 4, 5
DetailedPos (*C++ struct*), 5
DetailedPos::DetailedPos (*C++ function*), 5
DetailedPos::link1 (*C++ member*), 5
DetailedPos::link2 (*C++ member*), 5
DetailedPos::origin (*C++ member*), 5
DetailedPos::tool (*C++ member*), 5
DifferentialController (*C++ class*), 32
DifferentialController::DifferentialController (*C++ function*), 32, 34
DifferentialController::getAngOutput (*C++ function*), 33, 34
DifferentialController::getAngSetpoint (*C++ function*), 33, 34
DifferentialController::getAxleTrack (*C++ function*), 33, 34
DifferentialController::getLinOutput (*C++ function*), 33, 34
DifferentialController::getLinSetpoint (*C++ function*), 33, 34
DifferentialController::load (*C++ function*), 34
DifferentialController::m_angInput (*C++ member*), 35
DifferentialController::m_angPID (*C++ member*), 35
DifferentialController::m_angSetpoint (*C++ member*), 35
DifferentialController::m_angVelOutput (*C++ member*), 35
DifferentialController::m_axleTrack (*C++ member*), 35
DifferentialController::m_leftWheel (*C++ member*), 35
DifferentialController::m_linInput (*C++ member*), 35
DifferentialController::m_linPID (*C++ member*), 35
DifferentialController::m_linSetpoint (*C++ member*), 35
DifferentialController::m_linVelOutput (*C++ member*), 35
DifferentialController::m_mutex (*C++ member*), 35
DifferentialController::m_rightWheel (*C++ member*), 35
DifferentialController::onProcessEnabling (*C++ function*), 34, 35
DifferentialController::process (*C++ function*), 34
DifferentialController::save (*C++ function*), 34
DifferentialController::setAxleTrack (*C++ function*), 33, 34
DifferentialController::setInputs (*C++ function*), 32, 34
DifferentialController::setPID (*C++ function*), 33, 34
DifferentialController::setSetpoints (*C++ function*), 32, 34
DifferentialController::setWheels (*C++ function*), 33, 34
Dynamixel (*C++ member*), 93, 94
DynamixelClass (*C++ class*), 36

DynamixelClass::action (C++ function), 36, 37
 DynamixelClass::begin (C++ function), 36, 37
 DynamixelClass::end (C++ function), 36, 37
 DynamixelClass::ledStatus (C++ function), 37, 38
 DynamixelClass::lockRegister (C++ function), 36, 37
 DynamixelClass::move (C++ function), 36, 37
 DynamixelClass::moveRW (C++ function), 36, 37
 DynamixelClass::moveSpeed (C++ function), 36, 37
 DynamixelClass::moveSpeedRW (C++ function), 36, 37
 DynamixelClass::moving (C++ function), 36, 37
 DynamixelClass::ping (C++ function), 36, 37
 DynamixelClass::readLoad (C++ function), 37, 38
 DynamixelClass::readPosition (C++ function), 36, 37
 DynamixelClass::readSpeed (C++ function), 37, 38
 DynamixelClass::readTemperature (C++ function), 36, 37
 DynamixelClass::readVoltage (C++ function), 36, 37
 DynamixelClass::reset (C++ function), 36, 37
 DynamixelClass::RWStatus (C++ function), 36, 37
 DynamixelClass::setAngleLimit (C++ function), 36, 37
 DynamixelClass::setBD (C++ function), 36, 37
 DynamixelClass::setCMargin (C++ function), 36, 37
 DynamixelClass::setCSlope (C++ function), 36, 37
 DynamixelClass::setEndless (C++ function), 36, 37
 DynamixelClass::setID (C++ function), 36, 37
 DynamixelClass::setLEDAAlarm (C++ function), 36, 37
 DynamixelClass::setMaxTorque (C++ function), 36, 37
 DynamixelClass::setMaxTorqueRAM (C++ function), 36, 37
 DynamixelClass::setPunch (C++ function), 36, 37
 DynamixelClass::setRDT (C++ function), 36, 37
 DynamixelClass::setShutdownAlarm (C++ function), 36, 37
 DynamixelClass::setSRL (C++ function), 36, 37
 DynamixelClass::setTempLimit (C++ function), 36, 37
 DynamixelClass::setVoltageLimit (C++ function), 36, 37

DynamixelClass::torqueStatus (C++ function), 37, 38
 DynamixelClass::turn (C++ function), 36, 37

E

EEPROM_SIZE (C macro), 140
 ENABLE_VELOCITYCONTROLLER_LOGS (C macro), 140
 EndStop (C++ class), 38
 EndStop::attach (C++ function), 38
 EndStop::detach (C++ function), 38
 EndStop::getState (C++ function), 38

F

FullSpeedServo (C++ class), 39
 FullSpeedServo::attach (C++ function), 39
 FullSpeedServo::attached (C++ function), 39
 FullSpeedServo::detach (C++ function), 39
 FullSpeedServo::process (C++ function), 40
 FullSpeedServo::read (C++ function), 39
 FullSpeedServo::SpeedWrite (C++ function), 39
 FullSpeedServo::write (C++ function), 39

I

IK::ArmManager (C++ class), 40
 IK::ArmManager::ArmManager (C++ function), 40
 IK::ArmManager::clip_Workspaceo_constraints (C++ function), 40
 IK::ArmManager::estimated_time_of_arrival (C++ function), 41
 IK::ArmManager::get_elbow (C++ function), 40
 IK::ArmManager::get_joints (C++ function), 40
 IK::ArmManager::get_link1 (C++ function), 40
 IK::ArmManager::get_link2 (C++ function), 40
 IK::ArmManager::get_link3 (C++ function), 40
 IK::ArmManager::get_origin (C++ function), 40
 IK::ArmManager::get_workspace_back (C++ function), 40
 IK::ArmManager::get_workspace_front (C++ function), 40
 IK::ArmManager::go_to (C++ function), 40
 IK::ArmManager::init (C++ function), 40
 IK::ArmManager::load (C++ function), 41
 IK::ArmManager::save (C++ function), 41
 IK::ArmManager::set_arm_link (C++ function), 40
 IK::ArmManager::set_initial_joint_pos (C++ function), 40
 IK::ArmManager::set_origin (C++ function), 40

IK::ArmManager::set_workspace (C++ function), 40
IK::ArmManager::workspace_center (C++ function), 40
IK::equals (C++ function), 85
IK::float_equals (C++ function), 85, 86
IK::Joint (C++ class), 41
IK::Joint::get_path (C++ function), 41
IK::Joint::Joint (C++ function), 41
IK::Joint::time_to_destination (C++ function), 41
IK::Matrix2 (C++ class), 41
IK::Matrix2::createMatrix21 (C++ function), 41
IK::Matrix2::createMatrix22 (C++ function), 41
IK::Matrix2::det (C++ function), 41
IK::Matrix2::free (C++ function), 41
IK::Matrix2::multMatrix22x12 (C++ function), 41
IK::Matrix2::norm (C++ function), 41
IK::Matrix2::solve (C++ function), 41
IK::Matrix3 (C++ class), 42
IK::Matrix3::createMatrix31 (C++ function), 42
IK::Matrix3::createMatrix33 (C++ function), 42
IK::Matrix3::det (C++ function), 42
IK::Matrix3::free (C++ function), 42
IK::Matrix3::multMatrix33x13 (C++ function), 42
IK::Matrix3::norm (C++ function), 42
IK::Matrix3::solve (C++ function), 42
IK::MotorWrapper (C++ class), 42
IK::MotorWrapper::arrived (C++ function), 43
IK::MotorWrapper::end (C++ function), 43
IK::MotorWrapper::getID (C++ function), 42
IK::MotorWrapper::getOFFSET (C++ function), 42
IK::MotorWrapper::init (C++ function), 42
IK::MotorWrapper::load (C++ function), 43
IK::MotorWrapper::MotorWrapper (C++ function), 42
IK::MotorWrapper::process (C++ function), 43
IK::MotorWrapper::save (C++ function), 43
IK::MotorWrapper::setGoalPos (C++ function), 42
IK::MotorWrapper::setID (C++ function), 42
IK::MotorWrapper::setOFFSET (C++ function), 42
IK::MotorWrapper::setVelocityProfile (C++ function), 42
IK::Picker (C++ class), 43
IK::Picker::_flip_elbow (C++ member), 44
IK::Picker::compute_jacobian (C++ function), 43
IK::Picker::flip_elbow (C++ function), 43
IK::Picker::forward_kinematics (C++ function), 43
IK::Picker::get_detailed_pos (C++ function), 43
IK::Picker::get_joints (C++ function), 43
IK::Picker::get_joints_vel (C++ function), 43
IK::Picker::get_path (C++ function), 43
IK::Picker::get_tool (C++ function), 43
IK::Picker::get_tool_vel (C++ function), 43
IK::Picker::init (C++ function), 43
IK::Picker::inverse_kinematics (C++ function), 43
IK::Picker::phi_axis (C++ member), 44
IK::Picker::synchronisation_time (C++ function), 43
IK::Picker::x_axis (C++ member), 44
IK::Picker::y_axis (C++ member), 44
IK::Scara (C++ class), 44
IK::Scara::compute_jacobian (C++ function), 44
IK::Scara::forward_kinematics (C++ function), 44
IK::Scara::get_detailed_pos (C++ function), 44
IK::Scara::get_joints (C++ function), 44
IK::Scara::get_joints_vel (C++ function), 44
IK::Scara::get_path (C++ function), 44
IK::Scara::get_tool (C++ function), 44
IK::Scara::get_tool_vel (C++ function), 44
IK::Scara::inverse_kinematics (C++ function), 44
IK::Scara::Scara (C++ function), 44
IK::Scara::synchronisation_time (C++ function), 44
IK::TrajectoryManager (C++ class), 45
IK::TrajectoryManager::get_timestep (C++ function), 45
IK::TrajectoryManager::init (C++ function), 45
IK::TrajectoryManager::is_arrived (C++ function), 45
IK::TrajectoryManager::load (C++ function), 45
IK::TrajectoryManager::move_directly (C++ function), 45
IK::TrajectoryManager::save (C++ function), 45
IK::TrajectoryManager::set_armManager (C++ function), 45
IK::TrajectoryManager::set_Motors (C++

function), 45
 IK::TrajectoryManager::set_timestep
 (C++ *function*), 45
 IK::TrajectoryManager::TrajectoryManager
 (C++ *function*), 45
 INPUT_VOLTAGE_ERROR (C *macro*), 142
 INPUT_VOLTAGE_ERROR_MASK (C *macro*), 142
 inrange (C++ *function*), 89, 90
 INSTRUCTION_ERROR (C *macro*), 142, 143
 INSTRUCTION_ERROR_MASK (C *macro*), 143

J

Joints (C++ *struct*), 6
 Joints::Joints (C++ *function*), 6
 Joints::th1 (C++ *member*), 6
 Joints::th2 (C++ *member*), 6
 Joints::th3 (C++ *member*), 6

L

LEFT (C *macro*), 143
 LOCK (C *macro*), 144

M

MANAGE_OPCODE (C *macro*), 145, 146
 Matrix (C++ *member*), 94
 matrix_t (C++ *type*), 172
 MatrixMath (C++ *class*), 45
 MatrixMath::Add (C++ *function*), 45
 MatrixMath::Copy (C++ *function*), 45
 MatrixMath::Invert (C++ *function*), 45
 MatrixMath::Multiply (C++ *function*), 45
 MatrixMath::Print (C++ *function*), 45
 MatrixMath::Scale (C++ *function*), 45
 MatrixMath::Subtract (C++ *function*), 45
 MatrixMath::Transpose (C++ *function*), 45
 MAX (C *macro*), 146
 MAX_JOINTS (C *macro*), 146
 MAX_NUM_OF_BATCHED_MOVES (C *macro*), 146
 MAX_PULSEWIDTH (C *macro*), 147
 MAX_VELOCITY (C *macro*), 147
 MIN (C *macro*), 147
 MIN_PULSEWIDTH (C *macro*), 147
 MIN_VELOCITY (C *macro*), 148
 Motor_state_t (C++ *struct*), 6
 Motor_state_t::err_code (C++ *member*), 6
 Motor_state_t::id (C++ *member*), 6
 Motor_state_t::timeout (C++ *member*), 6
 MoveBatch (C++ *class*), 46
 MoveBatch::addDuration (C++ *function*), 46
 MoveBatch::addMove (C++ *function*), 46
 MoveBatch::addVelocityProfile (C++ *function*), 46
 MoveBatch::batch (C++ *member*), 46
 MoveBatch::get_duration (C++ *function*), 46

MoveBatch::is_active (C++ *function*), 46
 MoveBatch::MoveBatch (C++ *function*), 46
 MoveCommand_t (C++ *struct*), 6
 MoveCommand_t::isActive (C++ *member*), 7
 MoveCommand_t::position (C++ *member*), 7
 MoveCommand_t::time (C++ *member*), 7
 MoveCommand_t::vel (C++ *member*), 7
 Mutex (C++ *class*), 46
 Mutex::acquire (C++ *function*), 46
 Mutex::Mutex (C++ *function*), 46
 Mutex::release (C++ *function*), 46

N

NonCopyable (C++ *class*), 47
 NonCopyable::NonCopyable (C++ *function*), 47

O

Odometry (C++ *class*), 47
 Odometry::getAngVel (C++ *function*), 48, 49
 Odometry::getAxleTrack (C++ *function*), 48, 49
 Odometry::getLinVel (C++ *function*), 48, 49
 Odometry::getPosition (C++ *function*), 48, 49
 Odometry::getSlippage (C++ *function*), 48, 49
 Odometry::load (C++ *function*), 49
 Odometry::m_angVel (C++ *member*), 50
 Odometry::m_axleTrack (C++ *member*), 50
 Odometry::m_leftCodewheel (C++ *member*), 50
 Odometry::m_linVel (C++ *member*), 50
 Odometry::m_mutex (C++ *member*), 50
 Odometry::m_pos (C++ *member*), 50
 Odometry::m_rightCodewheel (C++ *member*), 50
 Odometry::m_slippage (C++ *member*), 50
 Odometry::process (C++ *function*), 49
 Odometry::save (C++ *function*), 49
 Odometry::setAxleTrack (C++ *function*), 48, 49
 Odometry::setCodewheels (C++ *function*), 48, 49
 Odometry::setPosition (C++ *function*), 47, 49
 Odometry::setSlippage (C++ *function*), 48, 49
 OFF (C *macro*), 148
 ON (C *macro*), 148, 149
 OVERLOAD_ERROR (C *macro*), 149
 OVERLOAD_ERROR_MASK (C *macro*), 149

P

P_MM (C *macro*), 150
 path_t (C++ *struct*), 7
 path_t::path_th1 (C++ *member*), 7
 path_t::path_th2 (C++ *member*), 7
 path_t::path_th3 (C++ *member*), 7
 path_t::pos (C++ *member*), 7
 periodicmod (C++ *function*), 90, 91
 PeriodicProcess (C++ *class*), 51

PeriodicProcess::~~PeriodicProcess (C++ function), 51

PeriodicProcess::disable (C++ function), 51, 52

PeriodicProcess::enable (C++ function), 51

PeriodicProcess::getTimestep (C++ function), 51, 52

PeriodicProcess::isEnabled (C++ function), 51, 52

PeriodicProcess::onProcessDisabling (C++ function), 52

PeriodicProcess::onProcessEnabling (C++ function), 52

PeriodicProcess::process (C++ function), 52

PeriodicProcess::setTimestep (C++ function), 51, 52

PeriodicProcess::update (C++ function), 51, 52

PID (C++ class), 52

PID::compute (C++ function), 53, 54

PID::getKd (C++ function), 53, 55

PID::getKi (C++ function), 53, 55

PID::getKp (C++ function), 53, 55

PID::getMaxOutput (C++ function), 54, 55

PID::getMinOutput (C++ function), 53, 55

PID::load (C++ function), 54, 55

PID::PID (C++ function), 53, 54

PID::reset (C++ function), 53, 54

PID::save (C++ function), 54, 55

PID::setOutputLimits (C++ function), 53, 54

PID::setTunings (C++ function), 53, 54

PLAT (C macro), 150

Polynom (C++ struct), 7

Polynom::a0 (C++ member), 7

Polynom::a1 (C++ member), 7

Polynom::a2 (C++ member), 7

Polynom::Polynom (C++ function), 7

Pos (C++ struct), 8

Pos::max (C++ member), 8

Pos::min (C++ member), 8

Pos::Pos (C++ function), 8

Position (C++ struct), 8

Position::Position (C++ function), 8, 9

Position::theta (C++ member), 9

Position::x (C++ member), 9

Position::y (C++ member), 9

PositionController (C++ class), 56

PositionController::getAngPosThreshold (C++ function), 58, 59

PositionController::getAngVelKp (C++ function), 58, 59

PositionController::getAngVelMax (C++ function), 58, 59

PositionController::getAngVelSetpoint (C++ function), 57, 58

PositionController::getLinPosThreshold (C++ function), 58, 59

PositionController::getLinVelKp (C++ function), 57, 59

PositionController::getLinVelMax (C++ function), 58, 59

PositionController::getLinVelSetpoint (C++ function), 56, 58

PositionController::getPositionReached (C++ function), 57, 59

PositionController::load (C++ function), 58, 59

PositionController::PositionController (C++ function), 56, 58

PositionController::save (C++ function), 58, 59

PositionController::setMoveStrategy (C++ function), 57, 59

PositionController::setPosInput (C++ function), 56, 58

PositionController::setPosSetpoint (C++ function), 56, 58

PositionController::setPosThresholds (C++ function), 57, 59

PositionController::setThetaSetpoint (C++ function), 56, 58

PositionController::setVelLimits (C++ function), 57, 59

PositionController::setVelTunings (C++ function), 57, 58

PressureSensor (C++ class), 59

PressureSensor::change_pressure_threshold (C++ function), 59

PressureSensor::currentlyAtmospherePressure (C++ function), 59

PressureSensor::get_pressure_threshold (C++ function), 59

PressureSensor::getPressurekPa (C++ function), 59

PressureSensor::PressureSensor (C++ function), 59

PurePursuit (C++ class), 60

PurePursuit::addWaypoint (C++ function), 61, 62

PurePursuit::checkLookAheadGoal (C++ function), 63, 64

PurePursuit::checkProjectionGoal (C++ function), 63, 64

PurePursuit::computeVelSetpoints (C++ function), 63, 64

PurePursuit::Direction (C++ enum), 60

PurePursuit::Direction::BACKWARD (C++

enumerator), 60
 PurePursuit::Direction::FORWARD (C++ *enumerator*), 60
 PurePursuit::getDirection (C++ *function*), 61, 62
 PurePursuit::getDistAfterGoal (C++ *function*), 63, 64
 PurePursuit::getFinalAngle (C++ *function*), 61, 62
 PurePursuit::getLookAhead (C++ *function*), 62, 63
 PurePursuit::getLookAheadBis (C++ *function*), 62, 63
 PurePursuit::getNumWaypoints (C++ *function*), 62
 PurePursuit::getPositionReached (C++ *function*), 63, 64
 PurePursuit::getWaypoint (C++ *function*), 62
 PurePursuit::load (C++ *function*), 62, 63
 PurePursuit::m_direction (C++ *member*), 64
 PurePursuit::m_finalAngle (C++ *member*), 64
 PurePursuit::m_goalIndex (C++ *member*), 64
 PurePursuit::m_goalParam (C++ *member*), 64
 PurePursuit::m_goalReached (C++ *member*), 64
 PurePursuit::m_lookAhead (C++ *member*), 64
 PurePursuit::m_lookAheadBis (C++ *member*), 64
 PurePursuit::m_numWaypoints (C++ *member*), 64
 PurePursuit::m_waypoints (C++ *member*), 64
 PurePursuit::PurePursuit (C++ *function*), 61, 62
 PurePursuit::reset (C++ *function*), 61, 62
 PurePursuit::save (C++ *function*), 62, 63
 PurePursuit::setDirection (C++ *function*), 61, 62
 PurePursuit::setFinalAngle (C++ *function*), 61, 62
 PurePursuit::setLookAhead (C++ *function*), 61, 62
 PurePursuit::setLookAheadBis (C++ *function*), 61, 62
 PurePursuit::Waypoint (C++ *struct*), 9, 64
 PurePursuit::Waypoint::Waypoint (C++ *function*), 9, 65
 PurePursuit::Waypoint::x (C++ *member*), 9, 65
 PurePursuit::Waypoint::y (C++ *member*), 9, 65
 PUREPURSUIT_MAX_WAYPOINTS (C *macro*), 151

R

RANGE_ERROR (C *macro*), 151

RANGE_ERROR_MASK (C *macro*), 152
 RIGTH (C *macro*), 153
 RMP_TO_DEG_S (C *macro*), 153
 Rx_MODE (C *macro*), 153

S

saturate (C++ *function*), 91, 92
 Semaphore (C++ *class*), 65
 Semaphore::acquire (C++ *function*), 65
 Semaphore::release (C++ *function*), 65
 Semaphore::Semaphore (C++ *function*), 65
 Serializer (C++ *struct*), 10
 Serializer::buffer (C++ *member*), 11
 Serializer::operator<< (C++ *function*), 10
 Serializer::Serializer (C++ *function*), 10
 Serializer::write (C++ *function*), 10, 11
 SerialTalks (C++ *class*), 66
 SerialTalks::attach (C++ *function*), 66, 68
 SerialTalks::begin (C++ *function*), 66, 67
 SerialTalks::bind (C++ *function*), 66, 67
 SerialTalks::err (C++ *member*), 68
 SerialTalks::execinstruction (C++ *function*), 66, 68
 SerialTalks::execute (C++ *function*), 67, 68
 SerialTalks::generateRandomUUID (C++ *function*), 68
 SerialTalks::getSerializer (C++ *function*), 67, 68
 SerialTalks::getUUID (C++ *function*), 67, 68
 SerialTalks::Instruction (C++ *type*), 66
 SerialTalks::isConnected (C++ *function*), 67, 68
 SerialTalks::m_bytesCounter (C++ *member*), 70
 SerialTalks::m_bytesNumber (C++ *member*), 70
 SerialTalks::m_connected (C++ *member*), 70
 SerialTalks::m_crc (C++ *member*), 70
 SerialTalks::m_crc_tab (C++ *member*), 70
 SerialTalks::m_crc_tmp (C++ *member*), 70
 SerialTalks::m_crcBytesCounter (C++ *member*), 70
 SerialTalks::m_inputBuffer (C++ *member*), 70
 SerialTalks::m_instructions (C++ *member*), 70
 SerialTalks::m_lastRetcode (C++ *member*), 70
 SerialTalks::m_lastTime (C++ *member*), 70
 SerialTalks::m_mutex (C++ *member*), 70
 SerialTalks::m_order (C++ *member*), 70
 SerialTalks::m_outputBuffer (C++ *member*), 70

- SerialTalks::m_processings (C++ member), 70
- SerialTalks::m_state (C++ member), 70
- SerialTalks::m_stream (C++ member), 70
- SerialTalks::ostream (C++ class), 70, 72
- SerialTalks::ostream::begin (C++ function), 71, 73
- SerialTalks::ostream::m_parent (C++ member), 71, 73
- SerialTalks::ostream::m_retcode (C++ member), 71, 73
- SerialTalks::ostream::operator<< (C++ function), 71–73
- SerialTalks::ostream::write (C++ function), 71–73
- SerialTalks::out (C++ member), 68
- SerialTalks::Processing (C++ type), 66
- SerialTalks::receive (C++ function), 69
- SerialTalks::received_crc_value (C++ member), 70
- SerialTalks::send (C++ function), 67, 68
- SerialTalks::sendback (C++ function), 69
- SerialTalks::setUUID (C++ function), 67, 68
- SerialTalks::waitUntilConnected (C++ function), 67, 68
- SerialTalks::[anonymous] (C++ enum), 68, 69
- SerialTalks::[anonymous]::SERIALTALKS_CRC_RECEIVING_STATE (C++ enumerator), 68, 69
- SerialTalks::[anonymous]::SERIALTALKS_INTERRUPT_RECEIVING_STATE (C++ enumerator), 68, 69
- SerialTalks::[anonymous]::SERIALTALKS_INTERRUPT_STARTING_STATE (C++ enumerator), 68, 69
- SerialTalks::[anonymous]::SERIALTALKS_ORDER (C++ enumerator), 69
- SerialTalks::[anonymous]::SERIALTALKS_RESET (C++ enumerator), 69
- SerialTalks::[anonymous]::SERIALTALKS_WAITING_STATE (C++ enumerator), 68, 69
- SERIALTALKS_BAUDRATE (C macro), 154
- SERIALTALKS_CRC_SIZE (C macro), 155
- SERIALTALKS_DEFAULT_UUID_LENGTH (C macro), 155
- SERIALTALKS_DISCONNECT_OPCODE (C macro), 155, 156
- SERIALTALKS_FREE_BUFFER_OPCODE (C macro), 156
- SERIALTALKS_GETBUFFERSIZE_OPCODE (C macro), 156
- SERIALTALKS_GETEEPROM_OPCODE (C macro), 157
- SERIALTALKS_GETUUID_OPCODE (C macro), 157
- SERIALTALKS_INPUT_BUFFER_SIZE (C macro), 157, 158
- SERIALTALKS_MASTER_BYTE (C macro), 158
- SERIALTALKS_MAX_OPCODE (C macro), 158
- SERIALTALKS_MAX_PROCESSING (C macro), 159
- SERIALTALKS_OUTPUT_BUFFER_SIZE (C macro), 159
- SERIALTALKS_PING_OPCODE (C macro), 159, 160
- SERIALTALKS_RESEND_OPCODE (C macro), 160
- SERIALTALKS_RESEVED_OPCODE_0 (C macro), 160
- SERIALTALKS_RESEVED_OPCODE_1 (C macro), 161
- SERIALTALKS_RESEVED_OPCODE_2 (C macro), 161
- SERIALTALKS_RESEVED_OPCODE_3 (C macro), 161, 162
- SERIALTALKS_RESEVED_OPCODE_4 (C macro), 162
- SERIALTALKS_RESEVED_OPCODE_5 (C macro), 162
- SERIALTALKS_RESEVED_OPCODE_6 (C macro), 163
- SERIALTALKS_RESEVED_OPCODE_7 (C macro), 163
- SERIALTALKS_RESEVED_OPCODE_8 (C macro), 163, 164
- SERIALTALKS_RESEVED_OPCODE_9 (C macro), 164
- SERIALTALKS_SETEEPROM_OPCODE (C macro), 164
- SERIALTALKS_SETUUID_OPCODE (C macro), 165
- SERIALTALKS_SLAVE_BYTE (C macro), 165
- SERIALTALKS_STDERR_RETCODE (C macro), 165, 166
- SERIALTALKS_STDOUT_RETCODE (C macro), 166
- SERIALTALKS_UUID_ADDRESS (C macro), 166
- SERIALTALKS_UUID_LENGTH (C macro), 167
- SerialTopics (C++ class), 74
- SerialTopics::begin (C++ function), 74
- SerialTopics::bind (C++ function), 74
- SerialTopics::end (C++ function), 74
- SerialTopics::getSubscriptions (C++ function), 74
- SerialTopics::Subscription (C++ type), 74
- SerialTopics::subscription_t (C++ struct), 11, 74
- SerialTopics::subscription_t::enable (C++ member), 11, 75
- SerialTopics::subscription_t::func (C++ member), 11, 75
- SerialTopics::subscription_t::lasttime (C++ member), 11, 75
- SerialTopics::subscription_t::timestep (C++ member), 11, 75
- SERIALTOPICS_DEFAULT_TIMING (C macro), 167
- SERIALTOPICS_MAX_OPCODE (C macro), 167, 168
- ShiftRegister (C++ class), 75
- ShiftRegister::attach (C++ function), 75
- ShiftRegister::m_CLOCK (C++ member), 75
- ShiftRegister::m_DATA (C++ member), 75
- ShiftRegister::m_LATCH (C++ member), 75
- ShiftRegister::m_register (C++ member), 75
- ShiftRegister::SetHigh (C++ function), 75
- ShiftRegister::SetLow (C++ function), 75
- ShiftRegister::shift (C++ function), 75
- ShiftRegister::write (C++ function), 75

[sign \(C++ function\), 92, 93](#)
[STEP_BY_REV \(C macro\), 169](#)
[StepByStepMotor \(C++ class\), 76](#)
[StepByStepMotor::attach \(C++ function\), 76](#)
[StepByStepMotor::begin \(C++ function\), 76](#)
[StepByStepMotor::disable \(C++ function\), 76](#)
[StepByStepMotor::enable \(C++ function\), 76](#)
[StepByStepMotor::get_position \(C++ function\), 76](#)
[StepByStepMotor::get_speed \(C++ function\), 76](#)
[StepByStepMotor::set_position \(C++ function\), 76](#)
[StepByStepMotor::set_speed \(C++ function\), 76](#)
[StepByStepMotor::step \(C++ function\), 76](#)
[StepByStepMotor::StepByStepMotor \(C++ function\), 76](#)
[StepByStepMotor::update \(C++ function\), 76](#)
[String \(C++ type\), 173](#)
[SUBSCRIBE \(C macro\), 169](#)

T

[talks \(C++ member\), 95, 96](#)
[TaskManager \(C++ class\), 76](#)
[TaskManager::create_task \(C++ function\), 76](#)
[TaskManager::delete_task \(C++ function\), 76](#)
[TaskManager::task_is_running \(C++ function\), 76](#)
[TaskManager::TaskManager \(C++ function\), 76](#)
[Thread \(C++ class\), 77](#)
[Thread::join \(C++ function\), 77](#)
[Thread::kill \(C++ function\), 77](#)
[Thread::Thread \(C++ function\), 77](#)
[TIME_OUT \(C macro\), 170](#)
[topics \(C++ member\), 96, 97](#)
[TrajectoryTime \(C++ struct\), 12](#)
[TrajectoryTime::t1 \(C++ member\), 12](#)
[TrajectoryTime::t2 \(C++ member\), 12](#)
[TrajectoryTime::tf \(C++ member\), 12](#)
[TrajectoryTime::TrajectoryTime \(C++ function\), 12](#)
[TurnOnTheSpot \(C++ class\), 77](#)
[TurnOnTheSpot::computeVelSetpoints \(C++ function\), 78](#)
[TurnOnTheSpot::Direction \(C++ enum\), 77](#)
[TurnOnTheSpot::Direction::CLOCK \(C++ enumerator\), 77](#)
[TurnOnTheSpot::Direction::TRIG \(C++ enumerator\), 77](#)
[TurnOnTheSpot::getPositionReached \(C++ function\), 78](#)
[TurnOnTheSpot::setDirection \(C++ function\), 78](#)

[TurnOnTheSpot::TurnOnTheSpot \(C++ function\), 78](#)
[TX_DELAY_TIME \(C macro\), 170](#)
[Tx_MODE \(C macro\), 171](#)

U

[UNSUBSCRIBE \(C macro\), 171](#)

V

[VacumPump \(C++ class\), 79](#)
[VacumPump::startPump \(C++ function\), 79](#)
[VacumPump::startSluice \(C++ function\), 79](#)
[VacumPump::stopPump \(C++ function\), 79](#)
[VacumPump::stopSluice \(C++ function\), 79](#)
[VacumPump::VacumPump \(C++ function\), 79](#)
[vector_t \(C++ struct\), 12](#)
[vector_t::acc \(C++ member\), 12](#)
[vector_t::pos \(C++ member\), 12](#)
[vector_t::t \(C++ member\), 12](#)
[vector_t::vel \(C++ member\), 12](#)
[Vel \(C++ struct\), 12](#)
[Vel::max \(C++ member\), 13](#)
[Vel::min \(C++ member\), 13](#)
[Vel::Vel \(C++ function\), 13](#)
[VelocityController \(C++ class\), 79](#)
[VelocityController::genRampSetpoint \(C++ function\), 82, 83](#)
[VelocityController::getAngSpinGoal \(C++ function\), 80, 82](#)
[VelocityController::getLinSpinGoal \(C++ function\), 80, 82](#)
[VelocityController::getMaxAngAcc \(C++ function\), 80, 81](#)
[VelocityController::getMaxAngDec \(C++ function\), 80, 81](#)
[VelocityController::getMaxLinAcc \(C++ function\), 80, 81](#)
[VelocityController::getMaxLinDec \(C++ function\), 80, 81](#)
[VelocityController::getSpinShutdown \(C++ function\), 80, 82](#)
[VelocityController::load \(C++ function\), 80, 82](#)
[VelocityController::m_angSpinGoal \(C++ member\), 84](#)
[VelocityController::m_linSpinGoal \(C++ member\), 83](#)
[VelocityController::m_maxAngAcc \(C++ member\), 83](#)
[VelocityController::m_maxAngDec \(C++ member\), 83](#)
[VelocityController::m_maxLinAcc \(C++ member\), 83](#)

VelocityController::m_maxLinDec (C++
member), 83

VelocityController::m_rampAngVelSetpoint
(C++ member), 83

VelocityController::m_rampLinVelSetpoint
(C++ member), 83

VelocityController::m_spinShutdown (C++
member), 83

VelocityController::onProcessEnabling
(C++ function), 83

VelocityController::process (C++ function),
82, 83

VelocityController::save (C++ function), 81,
82

VelocityController::setMaxAngAcc (C++
function), 79, 81

VelocityController::setMaxAngDec (C++
function), 80, 81

VelocityController::setMaxLinAcc (C++
function), 80, 81

VelocityController::setMaxLinDec (C++
function), 80, 81

VelocityController::setSpinShutdown
(C++ function), 80, 81

VelocityController::VelocityController
(C++ function), 79, 81

VELOCITYCONTROLLER_LOGS_TIMESTEP (C
macro), 171, 172

VelocityControllerLogs (C++ class), 84

VelocityControllerLogs::m_controller
(C++ member), 84

VelocityControllerLogs::m_mutex (C++
member), 84

VelocityControllerLogs::process (C++
function), 84

VelocityControllerLogs::setController
(C++ function), 84

W

Workspace (C++ struct), 13

Workspace::elbow_or (C++ member), 13

Workspace::Workspace (C++ function), 13

Workspace::x_max (C++ member), 13

Workspace::x_min (C++ member), 13

Workspace::y_max (C++ member), 13

Workspace::y_min (C++ member), 13