|  |  |  |  |
| --- | --- | --- | --- |
|  | logo indesitnero     |  | | --- | |  | | |
| Authors: | Version:  Status: |

Disclaimer

La diffusione del presente documento è limitata a INDESIT COMPANY ed a …. ed è di proprietà di INDESIT COMPANY.

Ogni riproduzione da parte di altri soggetti senza esplicita autorizzazione di INDESIT COMPANY è pertanto vietata a norma delle leggi vigenti.

Strictly Confidential - All rights strictly reserved. Reproduction or issue to third parties in any form is not permitted without written authority from the proprietors.

Revision

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date** | **Version** | **Author** | **Reviser** | **Status** | **Notes** |
| 14-12-11 | 1.0 |  |  | Draft |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Table of contents

[Disclaimer 2](#_Toc314480677)

[Revision 3](#_Toc314480678)

[Table of contents 4](#_Toc314480679)

[1 Introduction 7](#_Toc314480680)

[1.1 Subject 7](#_Toc314480681)

[1.2 Scope 7](#_Toc314480682)

[1.3 Overview 7](#_Toc314480683)

[1.4 Terms 7](#_Toc314480684)

[2 Universal Motor 7](#_Toc314480685)

[2.1 Introduction 7](#_Toc314480686)

[2.2 Module Parameters 8](#_Toc314480687)

[2.2.1 Platform Parameters 8](#_Toc314480688)

[2.2.1.1 NO\_SIGNAL\_TIMEOUT\_VAL 8](#_Toc314480689)

[2.2.1.2 NO\_SIGNAL\_TIMEOUT\_MAX\_VAL 8](#_Toc314480690)

[2.2.2 Settings Parameters 8](#_Toc314480691)

[2.2.2.1 Definition of the pins: 8](#_Toc314480692)

[2.3 Public types definition 8](#_Toc314480693)

[2.3.1 PM\_UM\_IndexIDType 8](#_Toc314480694)

[2.3.2 PM\_UM\_InitData 9](#_Toc314480695)

[2.3.3 GIOM\_LoadStruct 9](#_Toc314480696)

[2.3.4 PM\_UM\_RpmType 9](#_Toc314480697)

[2.3.5 PM\_UM\_IndexMeasureType 9](#_Toc314480698)

[2.4 Interfaces 9](#_Toc314480699)

[2.4.1 Tachometer info 10](#_Toc314480700)

[2.4.2 Unbalance 10](#_Toc314480701)

[2.4.3 Gate info 10](#_Toc314480702)

[2.4.4 DrumInfo 10](#_Toc314480703)

[2.4.5 Measures 11](#_Toc314480704)

[2.5 APIs 11](#_Toc314480705)

[2.5.1 void PM\_UM\_InitSync(PM\_UM\_InitData \*pinlist ) 11](#_Toc314480706)

[2.5.2 void PM\_UM\_Tachometer\_Init (uint8\_t \_PinModule, bool\_t \_PullupEnabled) 11](#_Toc314480707)

[2.5.3 PM\_UM\_RpmType PM\_UM\_GetCurrentRpmSync(void) 11](#_Toc314480708)

[2.5.4 void PM\_UM\_SetTargetSync( GIOM\_LoadStruct \*cmd ) 11](#_Toc314480709)

[2.5.5 void PM\_UM\_StateMachine( void ) 11](#_Toc314480710)

[2.5.6 void PM\_UM\_Algorithm( void ) 11](#_Toc314480711)

[2.5.7 PM\_UM\_IndexMeasureType PM\_UM\_GetIndexResult (PM\_UM\_IndexIDType \_IndexID) 12](#_Toc314480712)

[2.5.8 PM\_UM\_IndexMeasureType PM\_UM\_GetIndexAverage (PM\_UM\_IndexIDType \_IndexID) 12](#_Toc314480713)

[2.5.9 bool\_t PM\_UM\_IsPhaseAngleMinAtSoftStart( void ) 12](#_Toc314480714)

[2.5.10 void PM\_UM\_ResetPhaseAngleMinAtSoftStart ( void ) 12](#_Toc314480715)

[2.5.11 uint16\_t PM\_UM\_GetCurrentPhaseAngle\_us(void) 12](#_Toc314480716)

[2.5.12 bool\_t PM\_UM\_IsTimeoutTachometer(void) 12](#_Toc314480717)

[2.5.13 uint16\_t PM\_UM\_GetUnbalance(void) 12](#_Toc314480718)

[2.6 Architectural description 12](#_Toc314480719)

[3 Tachometer 12](#_Toc314480720)

[This module provides the logic for the signal coming from tachometer. It relies on PWD driver. It also provides logic for timeout and out of range detection. 12](#_Toc314480721)

[3.1 Definitions and properties 12](#_Toc314480722)

[3.1.1 Poles 12](#_Toc314480723)

[3.1.2 Range 13](#_Toc314480724)

[3.2 Public Structures and types definitions 13](#_Toc314480725)

[3.3 Private Structures and types definitions 13](#_Toc314480726)

[3.3.1 Acquisition Mode 13](#_Toc314480727)

[3.3.2 State of Measurement 13](#_Toc314480728)

[3.3.3 Measurements 13](#_Toc314480729)

[3.4 Private functions 13](#_Toc314480730)

[3.4.1 IO\_ErrorType PM\_UM\_Tachometer\_SetAcqMode (PM\_UM\_TachoAcquisitionModeType \_AcqMode) 13](#_Toc314480731)

[3.4.2 void PM\_UM\_TachometerStartAcquisition(void) 13](#_Toc314480732)

[3.4.3 PM\_UM\_TachoMeasStateType PM\_UM\_TachometerGetMeasureState (void) 13](#_Toc314480733)

[3.4.4 PM\_UM\_TachoMeasRawValueType PM\_UM\_TachometerGetMeasureValue (void) 14](#_Toc314480734)

[3.4.5 void PM\_UM\_TachoTickToRPM(void) 14](#_Toc314480735)

[3.4.6 bool\_t PM\_UM\_TachoIsTimeout(void) 14](#_Toc314480736)

[3.4.7 void PM\_UM\_ResetTimeout(void) 14](#_Toc314480737)

[3.5 Public functions 14](#_Toc314480738)

[3.5.1 void PM\_UM\_TachoMonitor (void) 14](#_Toc314480739)

[4 Under discussion 14](#_Toc314480740)

[4.1 Average period measurement 14](#_Toc314480741)

# Introduction

## Subject

Platform 2 Motor control module

## Scope

Architecture.

## Overview

Chapter 2 describes the module and the interface.

## Terms

ICU – Input Capture Unit

OCU – Output Compare Unit

# Universal Motor

## Introduction

Il modulo permette di controllare un motore a corrente alternata ( 2 fasi ).

Il pilotaggio del motore è effettuato tramite un triac, la direzione del motore è controllata tramite due relè e la velocità è rilevata tramite un circuito che produce 8 impulsi ogni giro del motore.

La configurazione del modulo è effettuata dalla configurazione della piattaforma e dalla configurazione dei parametri di settings.

## Module Parameters

Control algorithm

& State machine

Loads

Inputs

Parameters

Partialized triac motor

(OCU)

2 relays for sense of rotation

(Digital)

Half field relay

(Digital)

Tachometer

(PWD)

Platform

### Platform Parameters

Descrizione dei parametri configurabili dalla piattaforma.

#### NO\_SIGNAL\_TIMEOUT\_VAL

No tacho signal timeout in ms.

#### NO\_SIGNAL\_TIMEOUT\_MAX\_VAL

Number of times that no tacho signal is detected continuously.

### Settings Parameters

Descrizione dei parametri configurabili tramite file di settings.

#### Definition of the pins:

* Gate of the triac
* Sense of rotation relays
* Half field relay

see 2.3.2 PM\_UM\_InitData

## Public types definition

### PM\_UM\_IndexIDType

enum PM\_UM\_IndexIDType

PM\_UM\_MEASURE\_PHASE

PM\_UM\_MEASURE\_ANTI\_FOAM

### PM\_UM\_InitData

typedef struct {

uint8\_t pintTriac;

uint8\_t pinClockwise;

uint8\_t pinAntiClockwise;

uint8\_t pinHalfField;

} PM\_UM\_InitData;

**pinTriac** pin of the partialized-triac. Must be a pin with OCU capability.

**pinClockWise** pinused to drive motor in AntiClockWise sense

**pinAntiClockWise** pinused to drive motor in AntiClockWise sense

**pinHalfField** pin used for half field

### GIOM\_LoadStruct

typedef struct

{

uint8\_t loadID;

L\_PARAM loadParam;

uint8\_t data[GIOM\_PARAMDATA];

} GIOM\_LoadStruct;

**LoadID** don’t care

**loadparam** don’t care

**data** array read as 16 bits array

|  |  |  |
| --- | --- | --- |
| Index | Description | um |
| **0** | **Target speed** | **rpm** |
| **1** | **Ramp time** | **10 ms** |
| **2** | **Flags**  **Bit0: last target up**  **Bit1: last target reached**  **Bit2: speed sign**  **Bit3: ZC ref not updated** |  |

### PM\_UM\_RpmType

int16\_t

### PM\_UM\_IndexMeasureType

uint16\_t

## Interfaces

The following structures are shared with external tools. They can be extended but not changed. Fields are not in a such order to optimize memory occupation.

### Tachometer info

|  |  |  |
| --- | --- | --- |
| Type | name | Description |
| uint32\_t | **period** | period of the signal in us (ex curr\_period\_tick) |
| uint16\_t | **noSignalTimeout** | No signal timeout in ms (Ex cnt\_no\_tach in 40 ms). Must be decremented by the period in ms of state machine when no tacho measure is available.  Must be set to NO\_SIGNAL\_TIMEOUT\_VAL |
| uint16\_t | **noSignalTimeoutCntr** | Counts the number of times that the timeout signal is detected (Ex noTachoInEnginePhaseOn\_Cntr).  When it reaches NO\_SIGNAL\_TIMEOUT\_MAX\_VAL the bit ????? is set to 1. |

### Unbalance

|  |  |  |
| --- | --- | --- |
| Type | name | Description |
| uint16\_t | **index** | Value as a measure of unbalance of the drum (ex measure) |

### Gate info

|  |  |  |
| --- | --- | --- |
| Type | name | Description |
| uint16\_t | **alpha** | Phase angle in us (ex alphan) |

### DrumInfo

|  |  |  |
| --- | --- | --- |
| Type | name | Description |
| int16\_t | **current\_speed** | current drum rpm: < 0 anticlokwise, > 0 clockwise |
| uint8\_t | **state** | State machine (ex curr\_phase) |
| uint16\_t | **absSpeedTarget** | Absolute value of the final target (ex v\_to) |
| uint16\_t | **timeTarget** | Ex t\_tgt |
| uint16\_t | **currenSpeedTarget** | Absolute value of the current target |
| uint16\_t | **options** | Ex cmdflags |
| uint8\_t | **softStartAlphaMinCnt** | Ex softStartAlphaMin\_cnt |
| uint16 | **MidFieldSpeed** | Mid field speed |
| uint8\_t | **MidFieldPwd** | Mid field password |

### Measures

|  |  |  |
| --- | --- | --- |
| Type | name | Description |
| uint32\_t | **accu** | accumulator (ex sum\_alpha) |
| Uint16\_t | **avg** | Average value |
| uint8\_t | **samples** | Number of samples |
| uint8\_t | **enabled** | State of the measure |
| uint8\_t | **result** | result |
| uint8\_t | **bound** |  |

## APIs

### void PM\_UM\_InitSync(PM\_UM\_InitData \*pinlist )

Module initialization. Called by LIB\_IOM\_InputInitialization().

### void PM\_UM\_Tachometer\_Init (uint8\_t \_PinModule, bool\_t \_PullupEnabled)

Called by LIB\_IOM\_InputInitialization() at processing of the input TACHO\_I. It :

* configures the PWD driver;
* enables the pullup on the pin if required.

**\_PinModule** pin identifier of the module [in]

**\_PullupEnabled** indicates if pullup must be enabled [in]

### PM\_UM\_RpmType PM\_UM\_GetCurrentRpmSync(void)

Returns the current speed of the drum in rpm (signed)

### void PM\_UM\_SetTargetSync( GIOM\_LoadStruct \*cmd )

Sets the target: speed, time and other control flags.

### void PM\_UM\_StateMachine( void )

It’s execution is asynchronous respect to the driving:

* supervision of data coming from tachometer (timeouts)
* motor start up (sense of rotation)
* target speed generation (ramp)
* unbalance

It must be ensured that this function is called exactly once.

The period of the execution must be defined outside.

### void PM\_UM\_Algorithm( void )

It’s execution is synchronous respect to the driving

* Index computation
  + PM\_UM\_MEASURE\_PHASE
  + PM\_UM\_MEASURE\_ANTI\_FOAM
* Regulator Implementation:
* Soft start
* On

It must be ensured that this function is called exactly once.

### PM\_UM\_IndexMeasureType PM\_UM\_GetIndexResult (PM\_UM\_IndexIDType \_IndexID)

Returns the result value.

**\_IndexID** Index ID

### PM\_UM\_IndexMeasureType PM\_UM\_GetIndexAverage (PM\_UM\_IndexIDType \_IndexID)

Returns the average value.

**\_IndexID** Index ID

### bool\_t PM\_UM\_IsPhaseAngleMinAtSoftStart( void )

Returns if motor control is applying min.

### void PM\_UM\_ResetPhaseAngleMinAtSoftStart ( void )

### uint16\_t PM\_UM\_GetCurrentPhaseAngle\_us(void)

### bool\_t PM\_UM\_IsTimeoutTachometer(void)

### uint16\_t PM\_UM\_GetUnbalance(void)

## Architectural description

The module consists of many sub-modules:

1. Tachometer
2. Motor drive

# Tachometer

## This module provides the logic for the signal coming from tachometer. It relies on PWD driver. It also provides logic for timeout and out of range detection.

## Definitions and properties

### Poles

Needed for conversion to rpm

### Range

Measure in tick must be in a defined range.

## Public Structures and types definitions

PM\_UM\_RpmType as int16\_t

## Private Structures and types definitions

### Acquisition Mode

enum PM\_UM\_TachoAcquisitionModeType

PM\_UM\_TACHO\_ONE\_SHOT

PM\_UM\_TACHO\_CONTINUOUS

### State of Measurement

Enum PM\_UM\_TachoMeasStateType

PM\_UM\_TACHO\_MEASURE\_NEW : new measure is available

PM\_UM\_TACHO\_MEASURE\_WAITING: measure not yet available

### Measurements

PM\_UM\_TachoMeasRawValueType as uint32\_t

## Private functions

All functions are used inside Universal Motor Module.

### IO\_ErrorType PM\_UM\_Tachometer\_SetAcqMode (PM\_UM\_TachoAcquisitionModeType \_AcqMode)

Set the acquisition mode. Changing the mode on the fly is possible.

**\_AcqMode** specifies the mode of acquisition [in]

### void PM\_UM\_TachometerStartAcquisition(void)

Starts acquisition if it’s stopped.

### PM\_UM\_TachoMeasStateType PM\_UM\_TachometerGetMeasureState (void)

Returns if a new measure is available. Sets the state of the measure to PM\_UM\_TACHO\_MEASURE\_WAITING. It must be ensured that this function is called exactly once.

### PM\_UM\_TachoMeasRawValueType PM\_UM\_TachometerGetMeasureValue (void)

Returns the latest measure.

### void PM\_UM\_TachoTickToRPM(void)

### bool\_t PM\_UM\_TachoIsTimeout(void)

### void PM\_UM\_ResetTimeout(void)

## Public functions

### void PM\_UM\_TachoMonitor (void)

* checks timeout
* checks if measure is out of range.

Properties: timeout in tenth of ms.

# Under discussion

## Average period measurement

Average grid period must be in LIB