

Stepper motor controller

1.0

Generated by Doxygen 1.8.5

Sun Apr 19 2015 13:26:41

Contents

1	Todo List	1
2	Module Index	3
2.1	Modules	3
3	Data Structure Index	5
3.1	Data Structures	5
4	File Index	7
4.1	File List	7
5	Module Documentation	9
5.1	config Library	9
5.1.1	Detailed Description	10
5.1.2	Macro Definition Documentation	11
5.1.2.1	check_pin	11
5.1.2.2	CHECKPIN	11
5.1.2.3	clr_bit	11
5.1.2.4	clr_port	11
5.1.2.5	DDECAY	11
5.1.2.6	DDIR_485	11
5.1.2.7	DDIRECTION	11
5.1.2.8	DECAY	11
5.1.2.9	DENABLE_STEPPER	11
5.1.2.10	DFAULT	11
5.1.2.11	DHOME	11
5.1.2.12	DIR_485	12
5.1.2.13	DIRECTION	12
5.1.2.14	DMISO	12
5.1.2.15	DMODE0	12
5.1.2.16	DMODE1	12
5.1.2.17	DMODE2	12
5.1.2.18	DMOSI	12

5.1.2.19	DRXD	12
5.1.2.20	DSCL	12
5.1.2.21	DSTEP	12
5.1.2.22	DTXD	12
5.1.2.23	ENABLE_STEPPER	12
5.1.2.24	false	13
5.1.2.25	FAULT	13
5.1.2.26	HOME	13
5.1.2.27	IDECAY	13
5.1.2.28	IDIR_485	13
5.1.2.29	IDIRECTION	13
5.1.2.30	IENABLE_STEPPER	13
5.1.2.31	IFault	13
5.1.2.32	IHOME	13
5.1.2.33	IMISO	13
5.1.2.34	IMODE0	13
5.1.2.35	IMODE1	13
5.1.2.36	IMODE2	14
5.1.2.37	IMOSI	14
5.1.2.38	IRXD	14
5.1.2.39	ISCL	14
5.1.2.40	ISTEP	14
5.1.2.41	ITXD	14
5.1.2.42	MISO	14
5.1.2.43	MODE0	14
5.1.2.44	MODE1	14
5.1.2.45	MODE2	14
5.1.2.46	MOSI	14
5.1.2.47	PDECAY	14
5.1.2.48	PDIR_485	15
5.1.2.49	PDIRECTION	15
5.1.2.50	PENABLE_STEPPER	15
5.1.2.51	PFAULT	15
5.1.2.52	PHOME	15
5.1.2.53	PMISO	15
5.1.2.54	PMODE0	15
5.1.2.55	PMODE1	15
5.1.2.56	PMODE2	15
5.1.2.57	PMOSI	15
5.1.2.58	PRXD	15

5.1.2.59	PSCL	15
5.1.2.60	PSTEP	16
5.1.2.61	PTXD	16
5.1.2.62	RXD	16
5.1.2.63	SCL	16
5.1.2.64	set_as_input	16
5.1.2.65	set_as_output	16
5.1.2.66	set_bit	16
5.1.2.67	set_port	16
5.1.2.68	STEP	16
5.1.2.69	true	16
5.1.2.70	TXD	16
5.2	DRV8825 Library	17
5.2.1	Detailed Description	18
5.2.2	Enumeration Type Documentation	18
5.2.2.1	decay_type	18
5.2.2.2	mode_type	18
5.2.2.3	step_type	19
5.2.3	Function Documentation	19
5.2.3.1	CalWorkParam	19
5.2.3.2	Count_Step	20
5.2.3.3	Decay	21
5.2.3.4	Disabled_Stepper	22
5.2.3.5	Enable_Stepper	23
5.2.3.6	Get_Decay	23
5.2.3.7	Get_Mode	24
5.2.3.8	Get_Start	25
5.2.3.9	Get_Steps_revol	25
5.2.3.10	GetAcceleration	26
5.2.3.11	GetCurrentSpeed	26
5.2.3.12	GetMaxSpeed	26
5.2.3.13	GetMinSpeed	27
5.2.3.14	GetRealAcc	27
5.2.3.15	GetRealMaxSpeed	28
5.2.3.16	GetRealMinSpeed	29
5.2.3.17	HelperRtoT	29
5.2.3.18	HelperTtoR	30
5.2.3.19	Home	31
5.2.3.20	InitStepper	32
5.2.3.21	Mode	33

5.2.3.22	RealToTime	34
5.2.3.23	Set_Steps_revol	35
5.2.3.24	SetAcceleration	36
5.2.3.25	SetCurrentSpeed	37
5.2.3.26	SetMaxSpeed	37
5.2.3.27	SetMinSpeed	38
5.2.3.28	SetRealAcc	39
5.2.3.29	SetRealMaxSpeed	40
5.2.3.30	SetRealMinSpeed	40
5.2.3.31	Step	41
5.2.3.32	Stop_Motion_fast	42
5.2.3.33	Stop_Motion_normal	43
5.2.3.34	store	43
5.2.3.35	TimeToReal	44
5.2.3.36	Way_Speed	46
5.3	main Library	48
5.3.1	Detailed Description	48
5.3.2	Function Documentation	48
5.3.2.1	Init_Input_Output	48
5.3.2.2	test	49
5.4	messages Library	50
5.4.1	Detailed Description	50
5.4.2	Function Documentation	50
5.4.2.1	Acc	50
5.4.2.2	ADec	51
5.4.2.3	Errormssg	52
5.4.2.4	FDec	53
5.4.2.5	Fstep	53
5.4.2.6	HStep	54
5.4.2.7	M16Step	54
5.4.2.8	M32Step	55
5.4.2.9	M8Step	56
5.4.2.10	Maxspeed	56
5.4.2.11	Minspeed	57
5.4.2.12	Qstep	58
5.4.2.13	RAcc	59
5.4.2.14	RMaxSpd	60
5.4.2.15	RMinSpd	60
5.4.2.16	SDec	61
5.4.2.17	SptRevol	62

5.4.2.18	Stop	63
5.5	UART Library	64
5.5.1	Detailed Description	64
5.5.2	Macro Definition Documentation	64
5.5.2.1	USART_BAUD_SELECT	64
5.5.2.2	USART_BAUD_SELECT_DOUBLE_SPEED	65
5.5.2.3	VEOL	65
5.5.3	Function Documentation	65
5.5.3.1	usart_getchar	65
5.5.3.2	usart_init	66
5.5.3.3	usart_putchar	67
5.6	utils Library	70
5.6.1	Detailed Description	70
5.6.2	Function Documentation	70
5.6.2.1	ConvertASCItouint64	70
5.6.2.2	int_to_string	71
5.6.2.3	putstring	72
5.6.2.4	putstring_P	73
6	Data Structure Documentation	77
6.1	cmd_struct Struct Reference	77
6.1.1	Detailed Description	77
6.1.2	Field Documentation	77
6.1.2.1	id	78
6.1.2.2	name	78
6.2	Motion Struct Reference	78
6.2.1	Detailed Description	78
6.2.2	Field Documentation	79
6.2.2.1	CurrentSpeed	79
6.2.2.2	StepsAccel	79
6.2.2.3	StepsConstSpeed	79
6.2.2.4	StepsDeaccel	79
6.3	Motor_Parameters Struct Reference	79
6.3.1	Detailed Description	80
6.3.2	Field Documentation	80
6.3.2.1	Acceleration	80
6.3.2.2	MaxSpeed	80
6.3.2.3	MinSpeed	81
6.4	ring_buffer Struct Reference	81
6.4.1	Detailed Description	81

6.4.2	Field Documentation	82
6.4.2.1	data	82
6.4.2.2	fillcount	82
6.4.2.3	head	82
6.4.2.4	nlines	82
6.4.2.5	size	82
6.4.2.6	tail	82
7	File Documentation	83
7.1	config.h File Reference	83
7.2	drv_8825.c File Reference	85
7.2.1	Macro Definition Documentation	87
7.2.1.1	PERTMR	87
7.2.1.2	TIMERCONST	87
7.2.2	Typedef Documentation	87
7.2.2.1	Motor_Parameters	87
7.2.3	Function Documentation	87
7.2.3.1	ISR	87
7.2.4	Variable Documentation	88
7.2.4.1	ConstSpd	88
7.2.4.2	CurrentDecay	88
7.2.4.3	CurrentMode	88
7.2.4.4	ModeStart	88
7.2.4.5	Motion	88
7.2.4.6	RealSpeed	88
7.2.4.7	StepDir	88
7.2.4.8	StepsPerRev	89
7.2.4.9	TimeParam	89
7.2.4.10	WorkingParam	89
7.3	drv_8825.h File Reference	89
7.4	interpreter.c File Reference	91
7.4.1	Function Documentation	91
7.4.1.1	get_cmd_id	91
7.4.1.2	get_cmd_name	92
7.4.2	Variable Documentation	93
7.4.2.1	cmd_tbl	93
7.5	interpreter.h File Reference	93
7.5.1	Enumeration Type Documentation	94
7.5.1.1	anonymous enum	94
7.5.2	Function Documentation	95

7.5.2.1	get_cmd_id	95
7.5.2.2	get_cmd_name	96
7.6	main.c File Reference	97
7.6.1	Macro Definition Documentation	98
7.6.1.1	F_CPU	98
7.6.1.2	USART_BAUD_RATE	98
7.6.2	Function Documentation	98
7.6.2.1	main	98
7.6.3	Variable Documentation	105
7.6.3.1	usart_str	105
7.7	main.h File Reference	106
7.8	messages.c File Reference	106
7.9	messages.h File Reference	107
7.10	usart.c File Reference	108
7.10.1	Macro Definition Documentation	109
7.10.1.1	B1SIZE	109
7.10.1.2	BSIZE	109
7.10.2	Typedef Documentation	109
7.10.2.1	ring_buffer_t	109
7.10.3	Function Documentation	109
7.10.3.1	ISR	109
7.10.3.2	ISR	110
7.10.4	Variable Documentation	110
7.10.4.1	brx	110
7.10.4.2	btx	110
7.10.4.3	usart_rx	110
7.10.4.4	usart_tx	110
7.11	usart.h File Reference	111
7.12	usartm8.h File Reference	111
7.12.1	Macro Definition Documentation	112
7.12.1.1	DOR0	112
7.12.1.2	FE0	112
7.12.1.3	MPCM0	112
7.12.1.4	RXB80	112
7.12.1.5	RXC0	112
7.12.1.6	RXCIE0	113
7.12.1.7	RXEN0	113
7.12.1.8	TXB80	113
7.12.1.9	TXC0	113
7.12.1.10	TXCIE0	113

7.12.1.11 TXEN0	113
7.12.1.12 U2X0	113
7.12.1.13 UBRR0H	113
7.12.1.14 UBRR0L	113
7.12.1.15 UBRR0L	113
7.12.1.16 UCPOLO	113
7.12.1.17 UCSR0A	113
7.12.1.18 UCSR0B	114
7.12.1.19 UCSR0C	114
7.12.1.20 UCSZ00	114
7.12.1.21 UCSZ01	114
7.12.1.22 UCSZ02	114
7.12.1.23 UDR0	114
7.12.1.24 UDRE0	114
7.12.1.25 UDRIE0	114
7.12.1.26 UMSEL0	114
7.12.1.27 UPE0	114
7.12.1.28 UPM00	114
7.12.1.29 UPM01	114
7.12.1.30 URSEL0	115
7.12.1.31 URSEL0	115
7.12.1.32 USART0_RXC_vect	115
7.12.1.33 USART0_TXC_vect	115
7.12.1.34 USART0_UDRE_vect	115
7.12.1.35 USARTXXX_H_	115
7.12.1.36 USBS0	115
7.13 utils.c File Reference	115
7.13.1 Variable Documentation	116
7.13.1.1 buf	116
7.14 utils.h File Reference	116

Chapter 1

Todo List

Global [Home](#) (void)

it doesnt work as expected

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

config Library	9
DRV8825 Library	17
main Library	48
messages Library	50
UART Library	64
utils Library	70

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

cmd_struct	77
Motion	
This structure defines the current step execution status. This structure is used for the timer interrupting and in the comments for initializing the spin	78
Motor_Parameters	
This structure defines types for motor parameters This type is used for declaration of three structures that are for the motor parameters used for timer, the real parameters of the motor and the working parameters that have been already calculated according to the microstep mode . .	79
ring_buffer	81

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

config.h	83
drv_8825.c	85
drv_8825.h	89
interpreter.c	91
interpreter.h	93
main.c	97
main.h	106
messages.c	106
messages.h	107
usart.c	108
usart.h	111
usartm8.h	111
utils.c	115
utils.h	116

Chapter 5

Module Documentation

5.1 config Library

General macros for port manipulation and port naming.

Macros

- #define `check_pin`(PINx, PINxn) (PINx & (1 << PINxn))
- #define `CHECKPIN`(x, y) ((x&(1<<y))!=0?1:0)
- #define `clr_bit`(byte, bit) byte &= ~(1 << bit)
- #define `clr_port`(PORTx, PORTxn) PORTx &= ~(1 << PORTxn)
- #define `DDECAY` DDRC
- #define `DDIR_485` DDRB
- #define `DDIRECTION` DDRB
- #define `DECAY` PC1
- #define `DENABLE_STEPPER` DDRD
- #define `DFAULT` DDRB
- #define `DHOME` DDRC
- #define `DIR_485` PB0
- #define `DIRECTION` PB2
- #define `DMISO` DDRB
- #define `DMODE0` DDRC
- #define `DMODE1` DDRC
- #define `DMODE2` DDRC
- #define `DMOSI` DDRB
- #define `DRXD` DDRD
- #define `DSCL` DDRB
- #define `DSTEP` DDRC
- #define `DTXD` DDRD
- #define `ENABLE_STEPPER` PD5
- #define `false` 1!=1
- #define `FAULT` PB1
- #define `HOME` PC2
- #define `IDECAY` PINC
- #define `IDIR_485` PINB
- #define `IDIRECTION` PINB
- #define `IENABLE_STEPPER` PIND
- #define `IFault` PINB
- #define `IHOME` PINC

- #define IMISO PINB
- #define IMODE0 PINC
- #define IMODE1 PINC
- #define IMODE2 PINC
- #define IMOSI PINB
- #define IRXD PIND
- #define ISCL PINB
- #define ISTEP PINC
- #define ITXD PIND
- #define MISO PB4
- #define MODE0 PC3
- #define MODE1 PC4
- #define MODE2 PC5
- #define MOSI PB3
- #define PDECAY PORTC
- #define PDIR_485 PORTB
- #define PDIRECTION PORTB
- #define PENABLE_STEPPER PORTD
- #define PFAULT PORTB
- #define PHOME PORTC
- #define PMISO PORTB
- #define PMODE0 PORTC
- #define PMODE1 PORTC
- #define PMODE2 PORTC
- #define PMOSI PORTB
- #define PRXD PORTD
- #define PSCL PORTB
- #define PSTEP PORTC
- #define PTXD PORTD
- #define RXD PD0
- #define SCL PB5
- #define set_as_input(DDRx, DDxn) DDRx &= ~(1 << DDxn)
- #define set_as_output(DDRx, DDxn) DDRx |= (1 << DDxn)
- #define set_bit(byte, bit) byte |= (1 << bit)
- #define set_port(PORTx, PORTxn) PORTx |= (1 << PORTxn)
- #define STEP PC0
- #define true 1==1
- #define TXD PD1

5.1.1 Detailed Description

General macros for port manipulation and port naming.

```
#include <config.h>
```

Defines simple macros for setting bits and bytes for manipulating DRV8825

Author

Bilyana Borisova bibishte@gmail.com

5.1.2 Macro Definition Documentation

5.1.2.1 `#define check_pin(PINx, PINxn) (PINx & (1 << PINxn))`

Test whether bit `PINxn` in IO register `PINx` is set. This will return a 0 if the port pin is driven low, and non-zero if the pin is driven high.

Definition at line 72 of file `config.h`.

5.1.2.2 `#define CHECKPIN(x, y) ((x&(1<<y))!=0?1:0)`

Definition at line 74 of file `config.h`.

5.1.2.3 `#define clr_bit(byte, bit) byte &= ~(1 << bit)`

Definition at line 44 of file `config.h`.

5.1.2.4 `#define clr_port(PORTx, PORTxn) PORTx &= ~(1 << PORTxn)`

Write logic zero bit `PORTxn` in IO register `PORTx`. When the pin is configured as an output pin, the port pin is driven low (zero).

Definition at line 66 of file `config.h`.

5.1.2.5 `#define DDECAY DDRC`

Definition at line 134 of file `config.h`.

5.1.2.6 `#define DDIR_485 DDRB`

Definition at line 138 of file `config.h`.

5.1.2.7 `#define DDIRECTION DDRB`

Definition at line 137 of file `config.h`.

5.1.2.8 `#define DECAY PC1`

Definition at line 92 of file `config.h`.

5.1.2.9 `#define DENABLE_STEPPER DDRD`

Definition at line 129 of file `config.h`.

5.1.2.10 `#define DFAULT DDRB`

Definition at line 124 of file `config.h`.

5.1.2.11 `#define DHOME DDRC`

Definition at line 121 of file `config.h`.

5.1.2.12 #define DIR_485 PB0

Definition at line 96 of file config.h.

5.1.2.13 #define DIRECTION PB2

Definition at line 95 of file config.h.

5.1.2.14 #define DMISO DDRB

Definition at line 123 of file config.h.

5.1.2.15 #define DMODE0 DDRC

Definition at line 131 of file config.h.

5.1.2.16 #define DMODE1 DDRC

Definition at line 132 of file config.h.

5.1.2.17 #define DMODE2 DDRC

Definition at line 133 of file config.h.

5.1.2.18 #define DMOSI DDRB

Definition at line 136 of file config.h.

5.1.2.19 #define DRXD DDRD

Definition at line 120 of file config.h.

5.1.2.20 #define DSCL DDRB

Definition at line 122 of file config.h.

5.1.2.21 #define DSTEP DDRC

Definition at line 135 of file config.h.

5.1.2.22 #define DTXD DDRD

Definition at line 130 of file config.h.

5.1.2.23 #define ENABLE_STEPPER PD5

Definition at line 87 of file config.h.

5.1.2.24 #define false 1!=1

Definition at line 162 of file config.h.

5.1.2.25 #define FAULT PB1

Definition at line 82 of file config.h.

5.1.2.26 #define HOME PC2

Definition at line 79 of file config.h.

5.1.2.27 #define IDECAY PINC

Definition at line 155 of file config.h.

5.1.2.28 #define IDIR_485 PINB

Definition at line 159 of file config.h.

5.1.2.29 #define IDIRECTION PINB

Definition at line 158 of file config.h.

5.1.2.30 #define IENABLE_STEPPER PIND

Definition at line 150 of file config.h.

5.1.2.31 #define IFAULT PINB

Definition at line 146 of file config.h.

5.1.2.32 #define IHOME PINC

Definition at line 143 of file config.h.

5.1.2.33 #define IMISO PINB

Definition at line 145 of file config.h.

5.1.2.34 #define IMODE0 PINC

Definition at line 152 of file config.h.

5.1.2.35 #define IMODE1 PINC

Definition at line 153 of file config.h.

5.1.2.36 #define IMODE2 PINC

Definition at line 154 of file config.h.

5.1.2.37 #define IMOSI PINB

Definition at line 157 of file config.h.

5.1.2.38 #define IRXD PIND

Definition at line 142 of file config.h.

5.1.2.39 #define ISCL PINB

Definition at line 144 of file config.h.

5.1.2.40 #define ISTEP PINC

Definition at line 156 of file config.h.

5.1.2.41 #define ITXD PIND

Definition at line 151 of file config.h.

5.1.2.42 #define MISO PB4

Definition at line 81 of file config.h.

5.1.2.43 #define MODE0 PC3

Definition at line 89 of file config.h.

5.1.2.44 #define MODE1 PC4

Definition at line 90 of file config.h.

5.1.2.45 #define MODE2 PC5

Definition at line 91 of file config.h.

5.1.2.46 #define MOSI PB3

Definition at line 94 of file config.h.

5.1.2.47 #define PDECAY PORTC

Definition at line 113 of file config.h.

5.1.2.48 #define PDIR_485 PORTB

Definition at line 117 of file config.h.

5.1.2.49 #define PDIRECTION PORTB

Definition at line 116 of file config.h.

5.1.2.50 #define PENABLE_STEPPER PORTD

Definition at line 108 of file config.h.

5.1.2.51 #define PFAULT PORTB

Definition at line 103 of file config.h.

5.1.2.52 #define PHOME PORTC

Definition at line 100 of file config.h.

5.1.2.53 #define PMISO PORTB

Definition at line 102 of file config.h.

5.1.2.54 #define PMODE0 PORTC

Definition at line 110 of file config.h.

5.1.2.55 #define PMODE1 PORTC

Definition at line 111 of file config.h.

5.1.2.56 #define PMODE2 PORTC

Definition at line 112 of file config.h.

5.1.2.57 #define PMOSI PORTB

Definition at line 115 of file config.h.

5.1.2.58 #define PRXD PORTD

Definition at line 99 of file config.h.

5.1.2.59 #define PSCL PORTB

Definition at line 101 of file config.h.

5.1.2.60 `#define PSTEP PORTC`

Definition at line 114 of file config.h.

5.1.2.61 `#define PTXD PORTD`

Definition at line 109 of file config.h.

5.1.2.62 `#define RXD PD0`

Definition at line 78 of file config.h.

5.1.2.63 `#define SCL PB5`

Definition at line 80 of file config.h.

5.1.2.64 `#define set_as_input(DDRx, DDxn) DDRx &= ~(1 << DDxn)`

Write logical zero bit `DDxn` in IO register `DDRx`. This will configure `Pxn` as an input pin.

Definition at line 54 of file config.h.

5.1.2.65 `#define set_as_output(DDRx, DDxn) DDRx |= (1 << DDxn)`

Write logical one bit `DDxn` in IO register `DDRx`. This will configure `Pxn` as an output pin.

Definition at line 49 of file config.h.

5.1.2.66 `#define set_bit(byte, bit) byte |= (1 << bit)`

Definition at line 43 of file config.h.

5.1.2.67 `#define set_port(PORTx, PORTxn) PORTx |= (1 << PORTxn)`

Write logic one bit `PORTxn` in IO register `PORTx`. When the pin is configured as an output pin, the port pin is driven high (one).

Definition at line 60 of file config.h.

5.1.2.68 `#define STEP PC0`

Definition at line 93 of file config.h.

5.1.2.69 `#define true 1==1`

Definition at line 161 of file config.h.

5.1.2.70 `#define TXD PD1`

Definition at line 88 of file config.h.

5.2 DRV8825 Library

Driver for stepper motor controller based on DRV8825 by Texas Instruments.

Enumerations

- enum `decay_type` { `SLOW_DECAY`, `FAST_DECAY`, `AUTO_DECAY` }
Type for decay status.
- enum `mode_type` { `MODE_FULL_STEP`, `MODE_HALF_STEP`, `MODE_QUATER_STEP`, `MODE_8_MICROSTEP`, `MODE_16_MICROSTEP`, `MODE_32_MICROSTEP` }
Type for mode status.
- enum `step_type` { `STEP_COUNTER_CLOCKWISE`, `STEP_CLOCKWISE` }
Type for the direction status.

Functions

- void `CalWorkParam` (void)
- void `Count_Step` (`step_type` step, uint64_t step_count)
- void `Decay` (`decay_type` decay)
- void `Disabled_Stepper` (void)
- void `Enable_Stepper` (void)
- `decay_type` `Get_Decay` (void)
- `mode_type` `Get_Mode` (void)
- uint8_t `Get_Start` (void)
- uint16_t `Get_Steps_revol` (void)
- uint16_t `GetAcceleration` (void)
- uint16_t `GetCurrentSpeed` (void)
- uint16_t `GetMaxSpeed` (void)
- uint16_t `GetMinSpeed` (void)
- uint16_t `GetRealAcc` (void)
- uint16_t `GetRealMaxSpeed` (void)
- uint16_t `GetRealMinSpeed` (void)
- uint64_t `HelperRtoT` (uint64_t)
- uint64_t `HelperTtoR` (uint64_t)
- uint8_t `Home` (void)
- void `InitStepper` (void)
- void `Mode` (`mode_type` mode)
- void `RealToTime` (void)
- void `Set_Steps_revol` (uint16_t step_rev)
- void `SetAcceleration` (uint16_t accel)
- void `SetCurrentSpeed` (uint16_t spd)
- void `SetMaxSpeed` (uint16_t spd)
- void `SetMinSpeed` (uint16_t spd)
- void `SetRealAcc` (uint16_t realacc)
- void `SetRealMaxSpeed` (uint16_t maxspd)
- void `SetRealMinSpeed` (uint16_t minspd)
- void `Step` (`step_type` step)
- void `Stop_Motion_fast` (void)
- void `Stop_Motion_normal` (void)
- void `store` (void)
- void `TimeToReal` (void)
- void `Way_Speed` (`step_type` step)

5.2.1 Detailed Description

Driver for stepper motor controller based on DRV8825 by Texas Instruments.

```
#include <drv_8825.h>
```

This modul contains basic functions for controlling stepper motor with the Texas Instruments DRV8825

Note

Typical application based on DRV8825

Author

Bilyana Borisova bibishte@gmail.com

5.2.2 Enumeration Type Documentation

5.2.2.1 enum decay_type

Type for decay status.

Defines the status of the decay modes.

Enumerator

SLOW_DECAY Status for setting slow decay.

FAST_DECAY Status for setting fast decay.

AUTO_DECAY Status for setting auto decay.

Definition at line 52 of file drv_8825.h.

```
52         {SLOW_DECAY,      /**< Status for setting slow decay. */
53         FAST_DECAY,       /**< Status for setting fast decay. */
54         AUTO_DECAY        /**< Status for setting auto decay. */
55     } decay_type;
```

5.2.2.2 enum mode_type

Type for mode status.

Defines the status of the modes.

Enumerator

MODE_FULL_STEP Status for setting the full step mode.

MODE_HALF_STEP Status for setting the half step mode.

MODE_QUATER_STEP Status for setting the quater step mode.

MODE_8_MICROSTEP Status for setting the 8 microstep mode.

MODE_16_MICROSTEP Status for setting the 16 microstep mode.

MODE_32_MICROSTEP Status for setting the 32 microstep mode.

Definition at line 62 of file drv_8825.h.

```
62         {MODE_FULL_STEP,  /**< Status for setting the full step mode. */
63         MODE_HALF_STEP,   /**< Status for setting the half step mode. */
64         MODE_QUATER_STEP, /**< Status for setting the quater step mode. */
65         MODE_8_MICROSTEP, /**< Status for setting the 8 microstep mode. */
66         MODE_16_MICROSTEP, /**< Status for setting the 16 microstep mode. */
67         MODE_32_MICROSTEP /**< Status for setting the 32 microstep mode. */
68     } mode_type;
```

5.2.2.3 enum step_type

Type for the direction status.

Defines the direction:clockwise or counter clock wise.

Enumerator

STEP_COUNTER_CLOCKWISE Status for setting the clockwise direction.

STEP_CLOCKWISE Status for setting the counter clockwise direction.

Definition at line 75 of file drv_8825.h.

```

75         {STEP_COUNTER_CLOCKWISE,    /**< Status for setting the clockwise
      direction. */
76         STEP_CLOCKWISE              /**< Status for setting the counter clockwise
      direction. */
77     } step_type;
```

5.2.3 Function Documentation

5.2.3.1 void CalWorkParam (void)

Calculates the maximum, minimum and the acceleration for the different step modes.

Returns

void

Definition at line 723 of file drv_8825.c.

```

724 {
725     switch ( CurrentMode )
726     {
727         case MODE_FULL_STEP:
728             WorkingParam.MaxSpeed = TimeParam.
MaxSpeed;
729             WorkingParam.MinSpeed = TimeParam.
MinSpeed;
730             WorkingParam.Acceleration = TimeParam.
Acceleration;
731             break;
732
733         case MODE_HALF_STEP:
734             WorkingParam.MaxSpeed = TimeParam.
MaxSpeed >> 1;
735             WorkingParam.MinSpeed = TimeParam.
MinSpeed >> 1;
736             WorkingParam.Acceleration = TimeParam.
Acceleration >> 1;
737             break;
738
739
740         case MODE_QUATER_STEP:
741             WorkingParam.MaxSpeed = TimeParam.
MaxSpeed >> 2;
742             WorkingParam.MinSpeed = TimeParam.
MinSpeed >> 2;
743             WorkingParam.Acceleration = TimeParam.
Acceleration >> 2;
744             break;
745
746         case MODE_8_MICROSTEP:
747             WorkingParam.MaxSpeed = TimeParam.
MaxSpeed >> 3;
748             WorkingParam.MinSpeed = TimeParam.
MinSpeed >> 3;
749             WorkingParam.Acceleration = TimeParam.
Acceleration >> 3;
750             break;
751
752
753         case MODE_16_MICROSTEP:
754             WorkingParam.MaxSpeed = TimeParam.
```

```

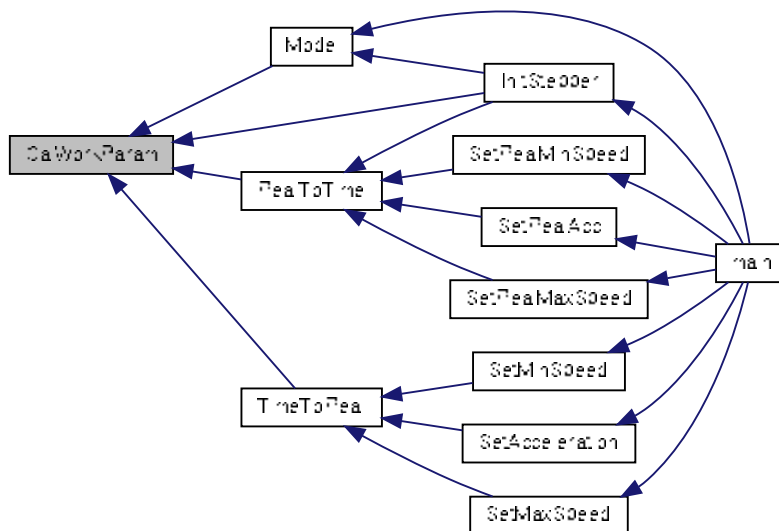
MaxSpeed >> 4;
755     WorkingParam.MinSpeed = TimeParam.
MinSpeed >> 4;
756     WorkingParam.Acceleration = TimeParam.
Acceleration >> 4;
757         break;
758
759
760     case MODE_32_MICROSTEP:
761         WorkingParam.MaxSpeed = TimeParam.
MaxSpeed >> 5;
762         WorkingParam.MinSpeed = TimeParam.
MinSpeed >> 5;
763         WorkingParam.Acceleration = TimeParam.
Acceleration >> 5;
764         break;
765
766     default:
767         Errormsg ( );
768
769         break;
770 }
771 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.2 void Count_Step (step_type step, uint64_t step_count)

Initializes the presented steps and calculates the steps for acceleration, deacceleration and the steps for constant speed

See Also

[step_type](#)

Parameters

<i>step</i>	the first argument.
<i>step_count</i>	the second argument.

Returns

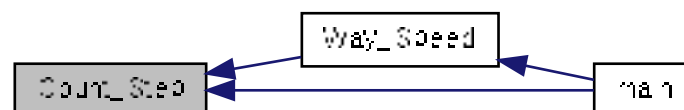
void

Definition at line 297 of file drv_8825.c.

```

298 {
299
300     uint16_t y;
301     StepDir = step;
302
303
304     y = ( WorkingParam.MinSpeed - WorkingParam.
MaxSpeed ) / WorkingParam.Acceleration;
305     y = y << 1;
306     if ( y < step_count )
307     {
308         Motion.StepsAccel = y >> 1;
309         Motion.StepsDeaccel = Motion.StepsAccel;
310         Motion.StepsConstSpeed = step_count - ( Motion.
StepsAccel + Motion.StepsDeaccel );
311
312     } else
313     {
314         Motion.StepsConstSpeed = 0;
315         Motion.StepsAccel = step_count >> 1;
316         Motion.StepsDeaccel = Motion.StepsAccel;
317         if ( ( step_count & 1 ) )
318         {
319             Motion.StepsAccel++;
320         }
321
322     }
323     clr_bit( TCCR1B, CS10 );
324     set_bit( TCCR1B, CS11 );
325     clr_bit( TCCR1B, CS12 );
326     TCNT1 = 65535 - WorkingParam.MinSpeed;
327     Motion.CurrentSpeed = WorkingParam.MinSpeed;
328     set_bit( TIMSK, TOIE1 );
329
330 }
```

Here is the caller graph for this function:



5.2.3.3 void Decay (decay_type decay)

Function that sets the decay type.

See Also

[decay_type](#)

Parameters

<i>decay</i>	the first argument.
--------------	---------------------

Returns

void

Definition at line 185 of file drv_8825.c.

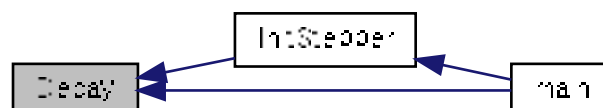
```

186 {
187     CurrentDecay = decay;
188     store( );
189     if ( decay == FAST_DECAY )
190     {
191         set_as_output( DDECAY, DECAY );
192         set_bit( PDECAY, DECAY );
193         return;
194     }
195     if ( decay == SLOW_DECAY )
196     {
197         set_as_output( DDECAY, DECAY );
198         clr_bit( PDECAY, DECAY );
199         return;
200     }
201     set_as_input( DDECAY, DECAY );
202     clr_bit( PDECAY, DECAY );
203 }
204
205 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.4 void Disabled_Stepper (void)

Function that disables the stepper motor.

Returns

void

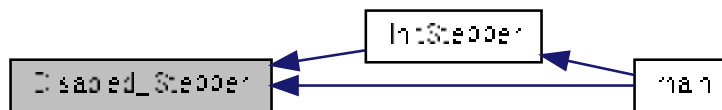
Definition at line 173 of file drv_8825.c.

```

174 {
175     set_bit( PENABLE_STEPPER, ENABLE_STEPPER );
176     ModeStart = 0;
177 }

```

Here is the caller graph for this function:

**5.2.3.5 void Enable_Stepper (void)**

Function that enables the stepper motor.

Returns

void

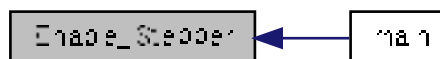
Definition at line 163 of file drv_8825.c.

```

164 {
165     clr_bit( PENABLE_STEPPER, ENABLE_STEPPER );
166     ModeStart = 1;
167 }

```

Here is the caller graph for this function:

**5.2.3.6 decay_type Get_Decay (void)**

Reads the setted decay.

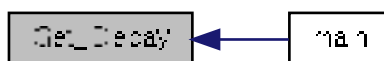
See Also[decay_type](#)**Returns**

decay_type

Definition at line 395 of file drv_8825.c.

```
396 {  
397     return CurrentDecay;  
398 }
```

Here is the caller graph for this function:

**5.2.3.7 mode_type Get_Mode (void)**

Reads the setted mode.

See Also[mode_type](#)**Returns**

mode_type

Definition at line 405 of file drv_8825.c.

```
406 {  
407     return CurrentMode;  
408 }
```

Here is the caller graph for this function:



5.2.3.8 uint8_t Get_Start (void)

Returns the mode.

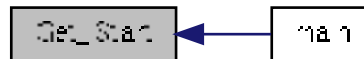
Returns

uint8_t

Definition at line 414 of file drv_8825.c.

```
415 {  
416     return ModeStart;  
417 }
```

Here is the caller graph for this function:



5.2.3.9 uint16_t Get_Steps_rev (void)

Gets the steps per revolution.

See Also

[StepsPerRev](#)

Returns

uint16_t

Definition at line 545 of file drv_8825.c.

```
546 {  
547     return StepsPerRev;  
548 }
```

Here is the caller graph for this function:



5.2.3.10 uint16_t GetAcceleration (void)

Reads the setted acceleration.

See Also

TimeParam.Acceleration

Returns

uint16_t

Definition at line 445 of file drv_8825.c.

```
446 {  
447     return TimeParam.Acceleration;  
448 }
```

Here is the caller graph for this function:



5.2.3.11 uint16_t GetCurrentSpeed (void)

Reads the current speed.

See Also

TimeParam.CurrentSpeed

Returns

uint16_t

Definition at line 456 of file drv_8825.c.

```
457 {  
458     return Motion.CurrentSpeed;  
459 }
```

5.2.3.12 uint16_t GetMaxSpeed (void)

Reads the setted maximum speed..

See Also

TimeParam.MaxSpeed

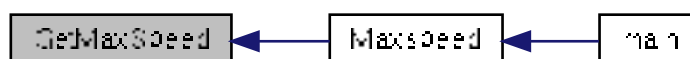
Returns

uint16_t

Definition at line 424 of file drv_8825.c.

```
425 {  
426     return TimeParam.MaxSpeed;  
427 }
```

Here is the caller graph for this function:

**5.2.3.13 uint16_t GetMinSpeed (void)**

Reads the setted minimum speed.

See Also

TimeParam.MinSpeed

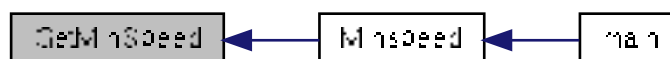
Returns

uint16_t

Definition at line 435 of file drv_8825.c.

```
436 {  
437     return TimeParam.MinSpeed;  
438 }
```

Here is the caller graph for this function:

**5.2.3.14 uint16_t GetRealAcc (void)**

Gets the real acceleration.

See Also

RealSpeed.Acceleration

Returns

uint16_t

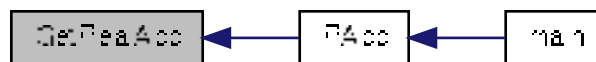
Definition at line 613 of file drv_8825.c.

```

614 {
615     return RealSpeed.Acceleration;
616 }

```

Here is the caller graph for this function:

**5.2.3.15 uint16_t GetRealMaxSpeed (void)**

Gets the real maximum speed.

See Also

RealSpeed.MaxSpeed

Returns

uint16_t

Definition at line 592 of file drv_8825.c.

```

593 {
594     return RealSpeed.MaxSpeed;
595 }

```

Here is the caller graph for this function:



5.2.3.16 uint16_t GetRealMinSpeed (void)

Gets the real minimum speed.

See Also

RealSpeed.MinSpeed

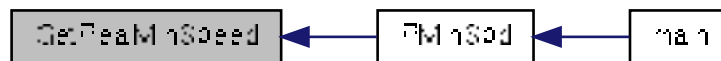
Returns

uint16_t

Definition at line 603 of file drv_8825.c.

```
604 {
605     return RealSpeed.MinSpeed;
606 }
```

Here is the caller graph for this function:



5.2.3.17 uint64_t HelperRtoT (uint64_t in)

Helper of the function RealToTime

See Also

[RealToTime](#)

Parameters

<i>in</i>	the first argument.
-----------	---------------------

Returns

uint64_t

Definition at line 838 of file drv_8825.c.

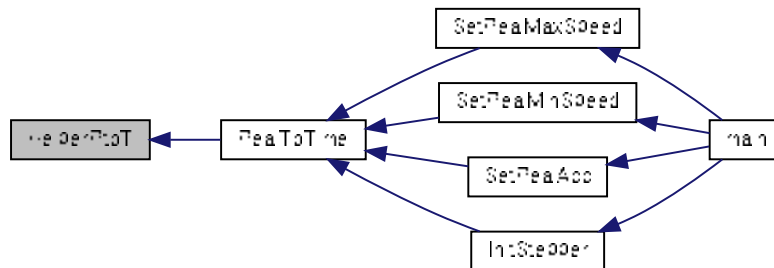
```
839 {
840     uint64_t revpermin;
841
842     /*
843      revpermin = in;
844      revpermin = revpermin*StepsPerRev;
845      revpermin = revpermin / 60;
846      revpermin = 1000000000 / revpermin;
847      revpermin = revpermin / PERTMR;
848     */
849     revpermin=TIMERCONST/(in*StepsPerRev);
850     if ( revpermin < 65500 )
851     {
852         return revpermin;
853     }
854 }
```

```

853     }
854     else
855     {
856         return 65500;
857     }
858 }

```

Here is the caller graph for this function:



5.2.3.18 uint64_t HelperTtoR (uint64_t in)

Helper of the function TimeToReal

See Also

[TimeToReal](#)

Parameters

<i>in</i>	the first argument.
-----------	---------------------

Returns

uint64_t

Definition at line 867 of file drv_8825.c.

```

868 {
869     uint64_t r;
870
871     //const uint64_t l=1000000000*60;
872
873     r=TIMERCONST/(in*StepsPerRev);
874
875     /*
876     r = in;
877
878
879     r = r * PERTMR;
880     r = r*StepsPerRev;
881     r = 10000000000 / r;
882     r = r * 6;
883
884     */
885
886
887
888
889     if ( r > 1 )
890     {
891         return r;

```

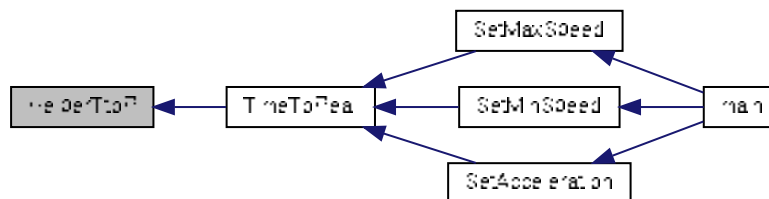


```

892     } else
893     {
894         return 1;
895         //putstring_P( PSTR( "Target MaxSpeed\r" ) );
896     }
897 }
898 }

```

Here is the caller graph for this function:



5.2.3.19 uint8_t Home (void)

The home of the motor.

Todo it doesnt work as expected

See Also

[step_type](#)

Returns

uint8_t

Definition at line 376 of file drv_8825.c.

```

377 {
378     int16_t i;
379
380     for ( i = 0; i < 128; i++ )
381     {
382         Step(STEP_COUNTER_CLOCKWISE);
383         if ( !(check_pin( IHOME, HOME )) )
384             return true;
385     }
386     return false;
387 }

```

Here is the call graph for this function:



5.2.3.20 void InitStepper (void)

Initialize the DRV232.

Returns

void

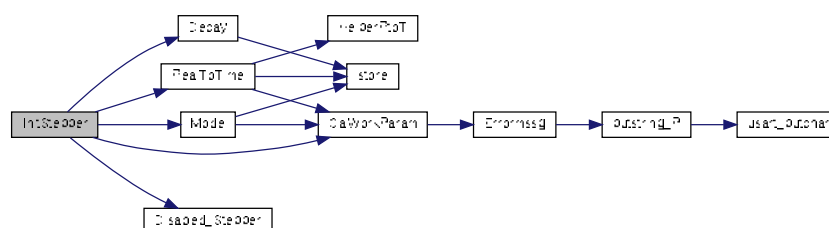
Definition at line 778 of file drv_8825.c.

```

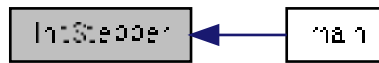
779 {
780     RealSpeed.MaxSpeed = eeprom_read_word( ( uint16_t * ) ( 0 ) );
781     RealSpeed.MinSpeed = eeprom_read_word( ( uint16_t * ) ( 2 ) );
782     RealSpeed.Acceleration = eeprom_read_word( ( uint16_t * ) ( 4 ) );
783     StepsPerRev = eeprom_read_word( ( uint16_t * ) ( 6 ) );
784     CurrentMode = eeprom_read_byte( ( uint8_t * ) ( 8 ) );
785     CurrentDecay = eeprom_read_byte( ( uint8_t * ) ( 9 ) );
786     if ( ( RealSpeed.MaxSpeed == 0 ) || ( RealSpeed.
MaxSpeed == 0xffff ) )
787     {
788         RealSpeed.MaxSpeed = 300;
789         Decay( AUTO_DECAY );
790         Mode( MODE_FULL_STEP );
791     }
792
793     if ( ( RealSpeed.MinSpeed == 0 ) || ( RealSpeed.
MinSpeed == 0xffff ) )
794     {
795         RealSpeed.MinSpeed = 10;
796     }
797
798     if ( ( RealSpeed.Acceleration == 0 ) || ( RealSpeed.
Acceleration == 0xffff ) )
799     {
800         RealSpeed.Acceleration = 100;
801     }
802
803     if ( ( StepsPerRev == 0 ) || ( StepsPerRev == 0xffff ) )
804     {
805         StepsPerRev = 400;
806     }
807
808     Disabled_Stepper( );
809     Decay( CurrentDecay );
810     Mode( CurrentMode );
811     RealToTime( );
812     CalWorkParam( );
813 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.21 void Mode (mode_type mode)

Function that sets the mode.

See Also

[mode_type](#)

Parameters

<i>mode</i>	the first argument.
-------------	---------------------

Returns

void

Definition at line 215 of file drv_8825.c.

```

216 {
217     CurrentMode = mode;
218     CalWorkParam( );
219     store( );
220
221     if ( mode == MODE_FULL_STEP )
222     {
223         clr_bit( PMODE0, MODE0 );
224         clr_bit( PMODE1, MODE1 );
225         clr_bit( PMODE2, MODE2 );
226     }
227
228     if ( mode == MODE_HALF_STEP )
229     {
230         set_bit( PMODE0, MODE0 );
231         clr_bit( PMODE1, MODE1 );
232         clr_bit( PMODE2, MODE2 );
233     }
234
235     if ( mode == MODE_QUATER_STEP )
236     {
237         clr_bit( PMODE0, MODE0 );
238         set_bit( PMODE1, MODE1 );
239         clr_bit( PMODE2, MODE2 );
240     }
241
242     if ( mode == MODE_8_MICROSTEP )
243     {
244         set_bit( PMODE0, MODE0 );
245         set_bit( PMODE1, MODE1 );
246         clr_bit( PMODE2, MODE2 );
247     }
248
249     if ( mode == MODE_16_MICROSTEP )
250     {
251         clr_bit( PMODE0, MODE0 );
252         clr_bit( PMODE1, MODE1 );
253         set_bit( PMODE2, MODE2 );
254     }
255

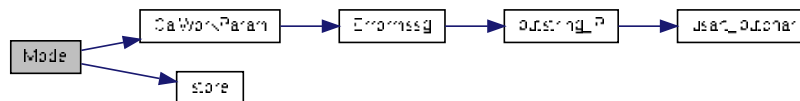
```

```

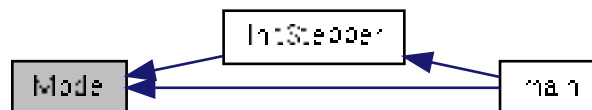
256     if ( mode == MODE_32_MICROSTEP )
257     {
258         set_bit( PMODE0, MODE0 );
259         clr_bit( PMODE1, MODE1 );
260         set_bit( PMODE2, MODE2 );
261     }
262 }
263 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.22 void RealToTime (void)

Calculates the real speed to the timer speed.

Returns

void

Definition at line 622 of file drv_8825.c.

```

623 {
624
625     uint16_t revpermin;
626     TimeParam.MaxSpeed=HelperRtoT(RealSpeed.
MaxSpeed);
627     TimeParam.MinSpeed=HelperRtoT(RealSpeed.
MinSpeed);
628
629     revpermin = ( TimeParam.MinSpeed - TimeParam.
MaxSpeed ) / RealSpeed.Acceleration;
630
631     //putstring_P( PSTR( "acc=" ) );
632     //putstring( int_to_string( ( uint64_t ) ( revpermin ) ) );
633     //putstring_P( PSTR( "\r" ) );
634
635     if ( revpermin < 1 )
636     {
637         TimeParam.Acceleration = 1;
638         // putstring_P( PSTR( "Acc low\r" ) );
639     } else
640     {

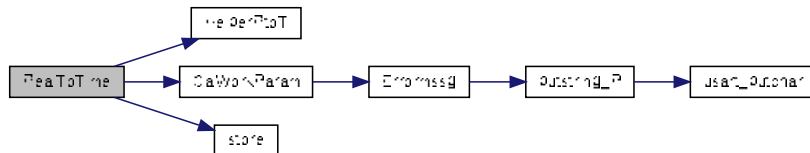
```

```

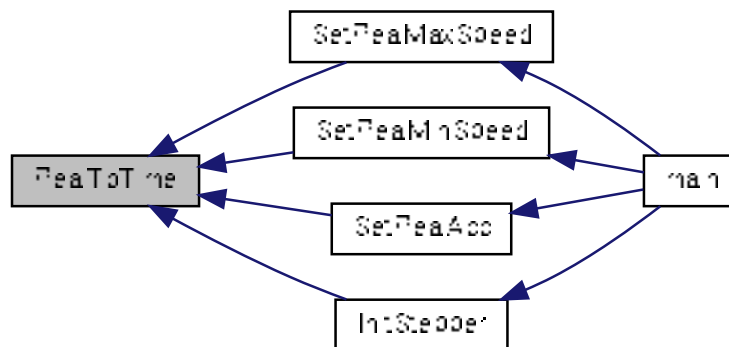
641     TimeParam.Acceleration = revpermin;
642 }
643
644 CalWorkParam( );
645 store( );
646 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.23 void Set_Steps_rev(uint16_t step_rev)

Sets the steps per revolution.

See Also

[StepsPerRev](#)

Parameters

<i>step_rev</i>	the first argument.
-----------------	---------------------

Returns

void

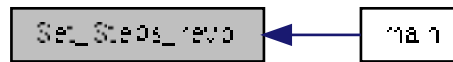
Definition at line 535 of file `drv_8825.c`.

```

536 {
537     StepsPerRev = step_rev;
538 }

```

Here is the caller graph for this function:



5.2.3.24 void SetAcceleration (uint16_t accel)

Function that sets the acceleration.

See Also

TimeParam.Acceleration

Parameters

<i>accel</i>	the first argument.
--------------	---------------------

Returns

void

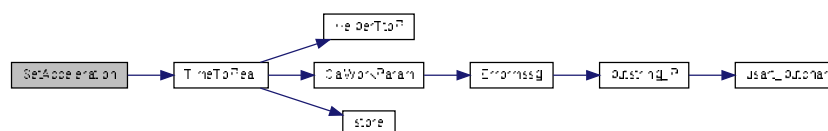
Definition at line 153 of file drv_8825.c.

```

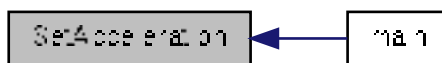
154 {
155     TimeParam.Acceleration = accel;
156     TimeToReal( );
157 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.25 void SetCurrentSpeed (uint16_t *spd*)

Sets the current speed.

See Also

[Motion.CurrentSpeed](#)

Parameters

<i>spd</i>	the first argument.
------------	---------------------

Returns

void

Definition at line 468 of file drv_8825.c.

```

469 {
470     Motion.CurrentSpeed = spd;
471 }
```

5.2.3.26 void SetMaxSpeed (uint16_t *spd*)

Function that sets the maximum speed.

See Also

[TimeParam.MaxSpeed](#)

Parameters

<i>spd</i>	the first argument.
------------	---------------------

Returns

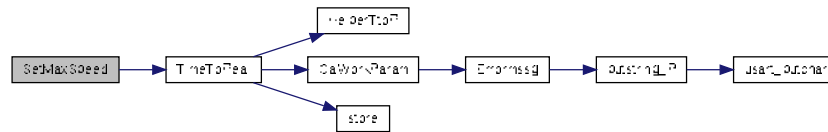
void

Definition at line 128 of file drv_8825.c.

```

129 {
130     TimeParam.MaxSpeed = spd;
131     TimeToReal( );
132 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.27 void SetMinSpeed (uint16_t spd)

Function that sets the minimum speed.

See Also

TimeParam.MinSpeed

Parameters

<i>spd</i>	the first argument.
------------	---------------------

Returns

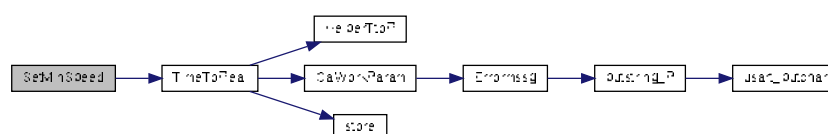
void

Definition at line 141 of file drv_8825.c.

```

142 {
143     TimeParam.MinSpeed = spd;
144     TimeToReal( );
145 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.28 void SetRealAcc (uint16_t *realacc*)

Sets the real acceleration.

See Also

RealSpeed.Acceleration

Parameters

<i>realacc</i>	the first argument.
----------------	---------------------

Returns

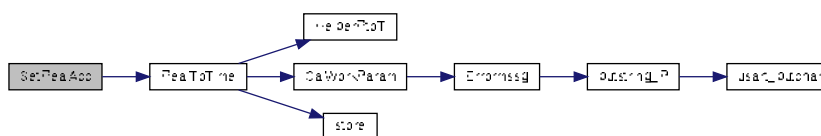
void

Definition at line 581 of file drv_8825.c.

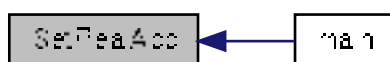
```

582 {
583     RealSpeed.Acceleration = realacc;
584     RealToTime ( );
585 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.29 void SetRealMaxSpeed (uint16_t maxspd)

Sets the real maximum speed.

See Also

RealSpeed.MaxSpeed

Parameters

<i>maxspd</i>	the first argument.
---------------	---------------------

Returns

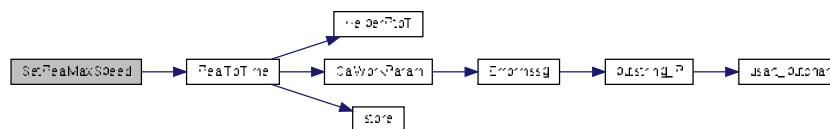
void

Definition at line 557 of file drv_8825.c.

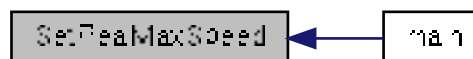
```

558 {
559     RealSpeed.MaxSpeed = maxspd;
560     RealToTime ( );
561 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.30 void SetRealMinSpeed (uint16_t minspd)

Sets the real minimum speed.

See Also

RealSpeed.MinSpeed

Parameters

<i>minspd</i>	the first argument.
---------------	---------------------

Returns

void

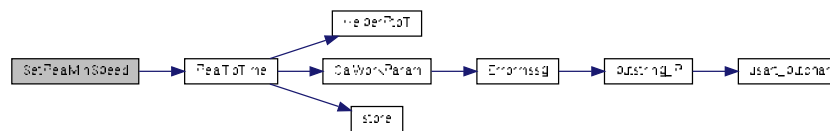
Definition at line 569 of file drv_8825.c.

```

570 {
571     RealSpeed.MinSpeed = minspd;
572     RealToTime ( );
573 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.31 void Step (step_type step)

Performs single step.

See Also

[step_type](#)

Parameters

<i>step</i>	the first argument.
-------------	---------------------

Returns

void

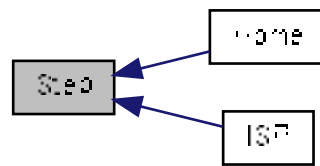
Definition at line 272 of file drv_8825.c.

```

273 {
274     if ( step == STEP_CLOCKWISE )
275     {
276         set_bit( PDIRECTION, DIRECTION );
277     }
278     else
279     {
280         clr_bit( PDIRECTION, DIRECTION );
281     }
282     set_bit( PSTEP, STEP );
283     _delay_us( 2 );
284     clr_bit( PSTEP, STEP );
285     _delay_us( 2 );
286 }

```

Here is the caller graph for this function:



5.2.3.32 void Stop_Motion_fast(void)

Stops the movement without deacceleration at the end.

Returns

void

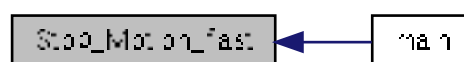
Definition at line 361 of file drv_8825.c.

```

362 {
363     ConstSpd = 0;
364     Motion.StepsAccel = 0;
365     Motion.StepsConstSpeed = 0;
366     Motion.StepsDeaccel = 0;
367 }

```

Here is the caller graph for this function:



5.2.3.33 void Stop_Motion_normal (void)

Stops the movement with deacceleration at the end.

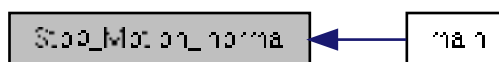
Returns

void

Definition at line 349 of file drv_8825.c.

```
350 {  
351     ConstSpd = 0;  
352     Motion.StepsAccel = 0;  
353     Motion.StepsConstSpeed = 0;  
354 }
```

Here is the caller graph for this function:



5.2.3.34 void store (void)

Calculates the real maximum speed, real minimum speed, real acceleration, steps per revolution, current mode and the current decay.

Returns

void

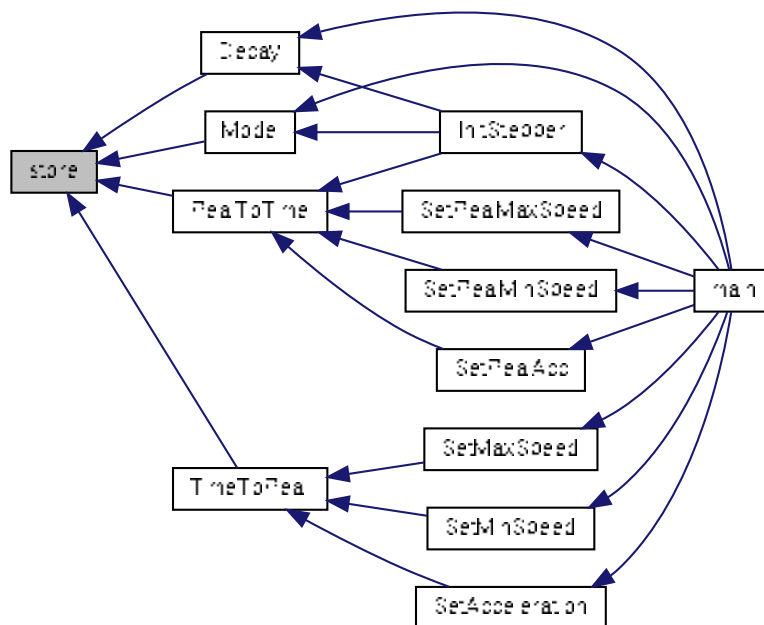
Definition at line 820 of file drv_8825.c.

```

821 {
822     eeprom_write_word( ( uint16_t * ) ( 0 ), RealSpeed.MaxSpeed );
823     eeprom_write_word( ( uint16_t * ) ( 2 ), RealSpeed.MinSpeed );
824     eeprom_write_word( ( uint16_t * ) ( 4 ), RealSpeed.Acceleration );
825     eeprom_write_word( ( uint16_t * ) ( 6 ), StepsPerRev );
826     eeprom_write_byte( ( uint8_t * ) ( 8 ), CurrentMode );
827     eeprom_write_byte( ( uint8_t * ) ( 9 ), CurrentDecay );
828
829 }
830 }

```

Here is the caller graph for this function:

**5.2.3.35 void TimeToReal (void)**

Calculates the timer speed to the real speed.

Returns

void

Definition at line 652 of file drv_8825.c.

```

653 {
654     uint16_t r;
655     RealSpeed.MaxSpeed=HelperTtoR(TimeParam.
656     MaxSpeed);
657     RealSpeed.MinSpeed=HelperTtoR(TimeParam.
658     MinSpeed);
659 }

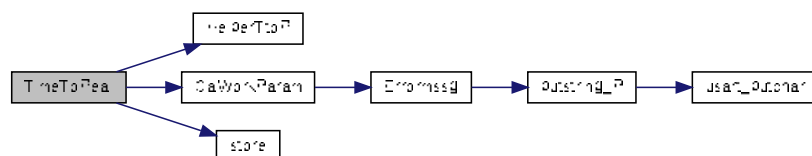
```

```

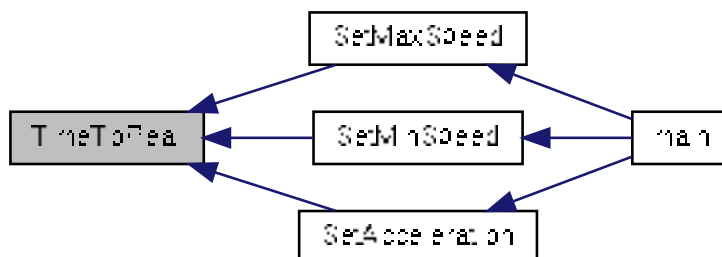
658 /*
659
660
661 //const uint64_t l=1000000000*60;
662
663 r = TimeParam.MaxSpeed;
664
665 r = r * PERTMR;
666 r = r*StepsPerRev;
667 r = 10000000000 / r;
668 r = r * 6;
669
670
671
672
673 if ( r > 1 )
674 {
675     RealSpeed.MaxSpeed = r;
676 } else
677 {
678     RealSpeed.MaxSpeed = 1;
679     //putstring_P( PSTR( "Target MaxSpeed\r" ) );
680 }
681
682
683 r = TimeParam.MinSpeed;
684 r = r * PERTMR;
685 r = r*StepsPerRev;
686 r = 60000000000 / r;
687 //r=r*6;
688
689
690 if ( r > 1 )
691 {
692     RealSpeed.MinSpeed = r;
693 } else
694 {
695     RealSpeed.MinSpeed = 1;
696     //putstring_P( PSTR( "Target Minspeed\r" ) );
697 }
698
699 */
700
701 r = ( TimeParam.MinSpeed - TimeParam.MaxSpeed ) /
TimeParam.Acceleration;
702
703 if ( r < 1 )
704 {
705     RealSpeed.Acceleration = 1;
706     //putstring_P( PSTR( "Acc low\r" ) );
707 } else
708 {
709     RealSpeed.Acceleration = r;
710 }
711
712 CalWorkParam( );
713 store( );
714 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.36 void Way_Speed (step_type step)

Movement with constant speed and constant rPM

See Also

[step_type](#)

Parameters

<i>step</i>	the first argument.
-------------	---------------------

Returns

void

Definition at line 339 of file `drv_8825.c`.

```

340 {
341     ConstSpd = 1;
342     Count_Step( step, 60000 );
343 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.3 main Library

This turns on the motor and directly goes in the continuous loop for controlling it.

Functions

- void `Init_Input_Output` (void)
- void `test` (void)

5.3.1 Detailed Description

This turns on the motor and directly goes in the continuous loop for controlling it.

```
#include <main.h>
```

This turns on the motor and directly goes in the continuous loop for controlling it.

Note

this is where all the magic happens

Author

Bilyana Borisova bibishte@gmail.com

5.3.2 Function Documentation

5.3.2.1 void Init_Input_Output (void)

Initial initialisation of input and output of microcontroller

Returns

void

Definition at line 561 of file main.c.

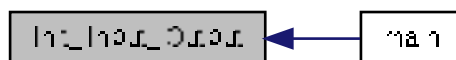
```
562 {
563     //input init
564
565     set_as_input( DRXD, RXD );
566     set_as_input( DHOME, HOME );
567     set_as_input( DSCL, SCL );
568     set_as_input( DMISO, MISO );
569     set_as_input( DFAULT, FAULT );
570
571     set_bit( PRXD, RXD );
572     set_bit( PSCL, SCL );
573     set_bit( PMISO, MISO );
574     set_bit( PHOME, HOME );
575
576     //output init
577
578     set_as_output( DENABLE_STEPPER, ENABLE_STEPPER );
579     set_as_output( DTXD, TXD );
580     set_as_output( DMODE0, MODE0 );
581     set_as_output( DMODE1, MODE1 );
582     set_as_output( DMODE2, MODE2 );
583     set_as_input( DDECAY, DECAY );
584     set_as_output( DSTEP, STEP );
585     set_as_output( DMOSI, MOSI );
586     set_as_output( DDIRECTION, DIRECTION );
587     set_as_output( DDIR_485, DIR_485 );
588 }
```

```

589     set_bit( PENABLE_STEPPER, ENABLE_STEPPER );
590     clr_bit( PMODE0, MODE0 );
591     clr_bit( PMODE1, MODE1 );
592     clr_bit( PMODE2, MODE2 );
593     clr_bit( PDECAY, DECAY );
594     clr_bit( PDIRECTION, DIRECTION );
595     set_bit( PDIR_485, DIR_485 );
596
597 }

```

Here is the caller graph for this function:



5.3.2.2 void test (void)

Just for testing

Returns

void

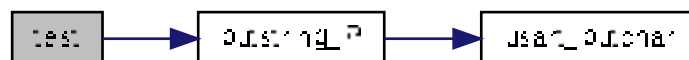
Definition at line 603 of file main.c.

```

604 {
605     putstring_P( PSTR( "Test?" ) );
606
607 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4 messages Library

This module contains functions that are printing messages.

Functions

- void [Acc](#) (void)
- void [ADec](#) (void)
- void [Errormssg](#) (void)
- void [FDec](#) (void)
- void [Fstep](#) (void)
- void [HStep](#) (void)
- void [M16Step](#) (void)
- void [M32Step](#) (void)
- void [M8Step](#) (void)
- void [Maxspeed](#) (void)
- void [Minspeed](#) (void)
- void [Qstep](#) (void)
- void [RAcc](#) (void)
- void [RMaxSpd](#) (void)
- void [RMinSpd](#) (void)
- void [SDec](#) (void)
- void [SptRevol](#) (void)
- void [Stop](#) (void)

5.4.1 Detailed Description

This module contains functions that are printing messages.

```
#include <messages.h>
```

This module contains functions that are printing stuff

Note

Typical functions that are saving me memory.

Author

Bilyana Borisova bibishte@gmail.com

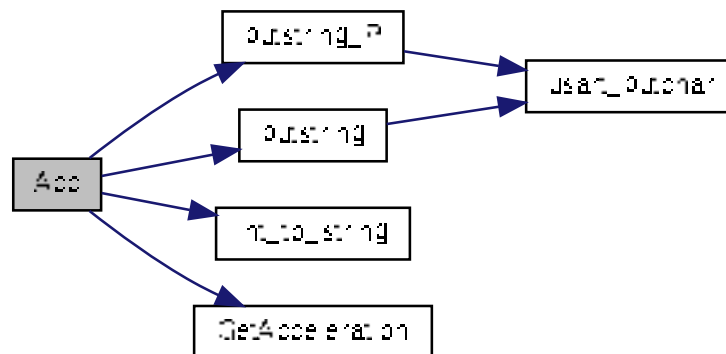
5.4.2 Function Documentation

5.4.2.1 void Acc (void)

Definition at line 77 of file messages.c.

```
78 {
79     putstring_P( PSTR( "Acc=" ) );
80     putstring( int_to_string( ( uint64_t ) (
      GetAcceleration( ) ) ) );
81     putstring_P( PSTR( "\r" ) );
82 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



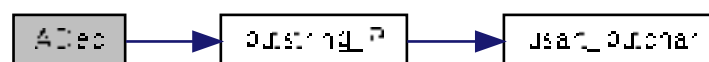
5.4.2.2 void ADec (void)

Definition at line 42 of file messages.c.

```

43 {
44     putstring_P( PSTR( "ADec\r" ) );
45 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



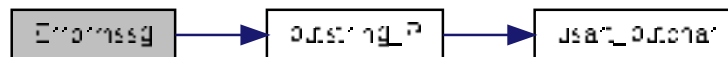
5.4.2.3 void Errormsg (void)

Definition at line 91 of file messages.c.

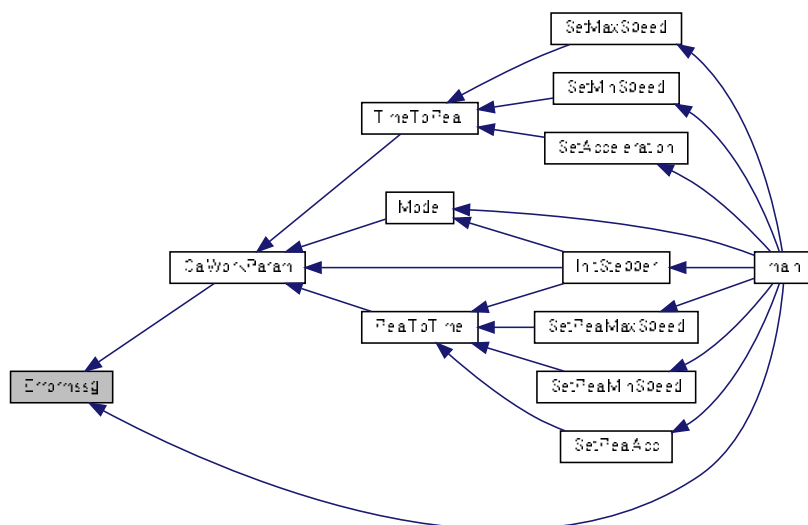
```

92 {
93     putstring_P( PSTR( "ERROR\r" ) );
94 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:

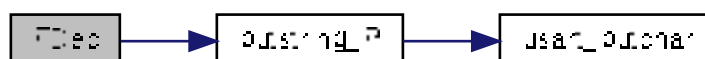


5.4.2.4 void FDec (void)

Definition at line 37 of file messages.c.

```
38 {  
39     putstring_P( PSTR( "FDec\r" ) );  
40 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

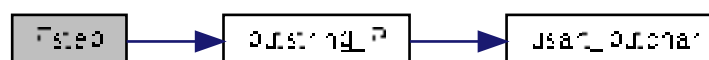


5.4.2.5 void Fstep (void)

Definition at line 47 of file messages.c.

```
48 {  
49     putstring_P( PSTR( "full step\r" ) );  
50 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



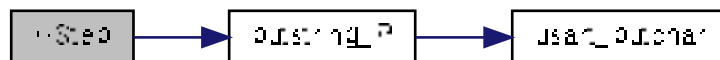
5.4.2.6 void HStep (void)

Definition at line 52 of file messages.c.

```

53 {
54     putstring_P( PSTR( "half step\r" ) );
55 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



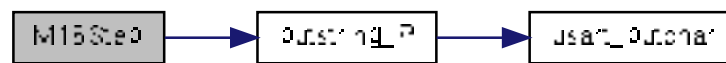
5.4.2.7 void M16Step (void)

Definition at line 67 of file messages.c.

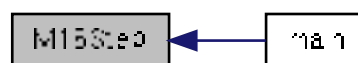
```

68 {
69     putstring_P( PSTR( "16 microsteps\r" ) );
70 }
  
```


Here is the call graph for this function:



Here is the caller graph for this function:



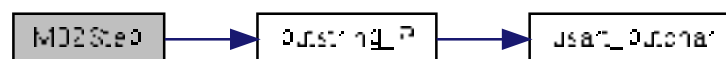
5.4.2.8 void M32Step (void)

Definition at line 72 of file messages.c.

```

73 {
74     putstring_P( PSTR( "32 microsteps\r" ) );
75 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.2.9 void M8Step (void)

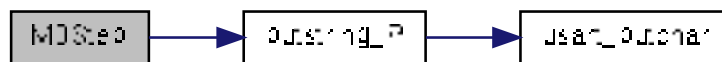
Definition at line 62 of file messages.c.

```

63 {
64     putstring_P( PSTR( "8 microsteps\r" ) );
65 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.2.10 void Maxspeed (void)

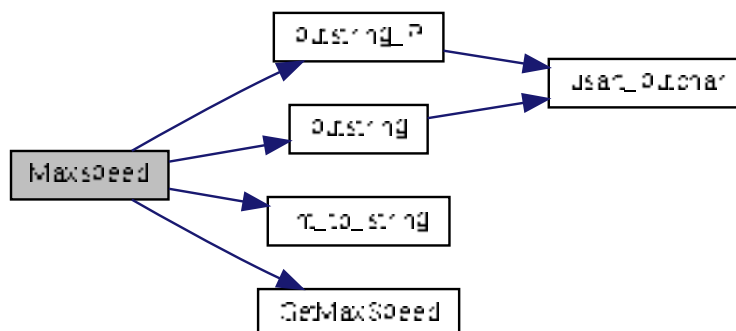
Definition at line 96 of file messages.c.

```

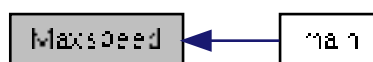
97 {
98     putstring_P( PSTR( "MaxSPD=" ) );
99     putstring( int_to_string( ( uint64_t ) ( GetMaxSpeed( ) ) ) );
100     putstring_P( PSTR( "\r" ) );
101 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.2.11 void Minspeed (void)

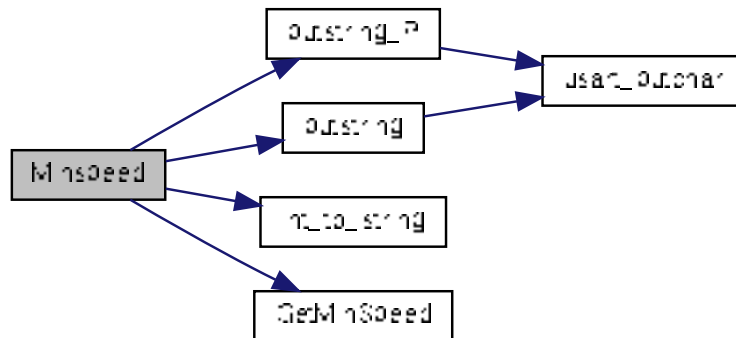
Definition at line 84 of file messages.c.

```

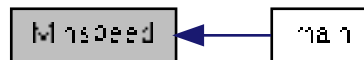
85 {
86     putstring_P( PSTR( "MinSPD=" ) );
87     putstring( int_to_string( ( uint64_t ) ( GetMinSpeed( ) ) ) );
88     putstring_P( PSTR( "\r" ) );
89 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



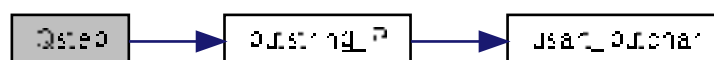
5.4.2.12 void Qstep (void)

Definition at line 57 of file messages.c.

```

58 {
59     putstring_P( PSTR( "quater step\r" ) );
60 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



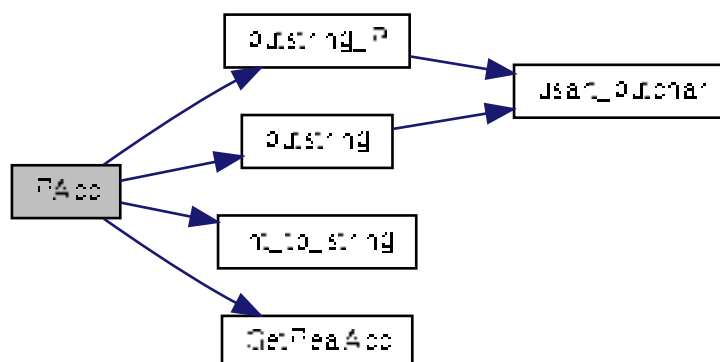
5.4.2.13 void RAcc (void)

Definition at line 124 of file messages.c.

```

125 {
126     putstring_P( PSTR( "RAcc=" ) );
127     putstring( int_to_string( ( uint16_t ) ( GetRealAcc( ) ) ) );
128     putstring_P( PSTR( "\r" ) );
129 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



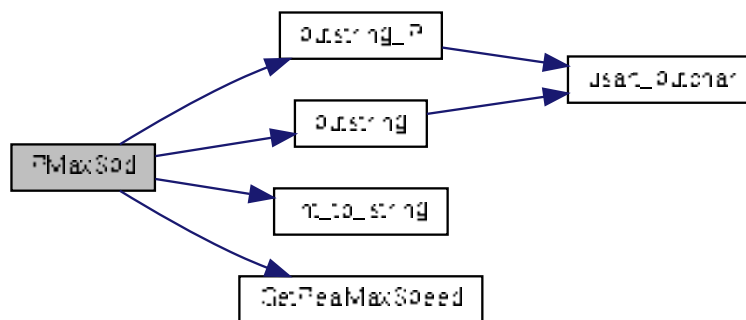
5.4.2.14 void RMaxSpd (void)

Definition at line 117 of file messages.c.

```

118 {
119     putstring_P( PSTR( "RMaxSPD=" ) );
120     putstring( int_to_string( ( uint64_t ) (
    GetRealMaxSpeed( ) ) ) );
121     putstring_P( PSTR( "\r" ) );
122 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



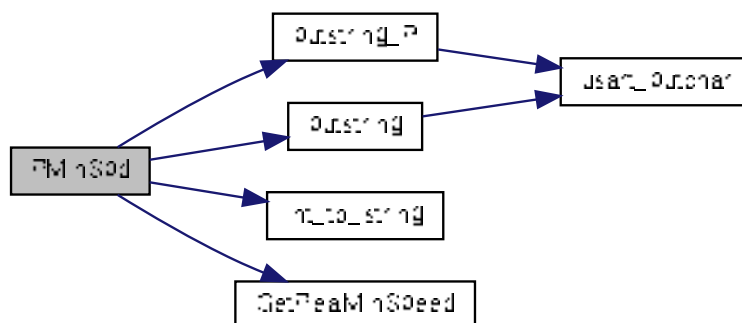
5.4.2.15 void RMinSpd (void)

Definition at line 110 of file messages.c.

```

111 {
112     putstring_P( PSTR( "RMinSpd=" ) );
113     putstring( int_to_string( ( uint16_t ) (
    GetRealMinSpeed( ) ) ) );
114     putstring_P( PSTR( "\r" ) );
115 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



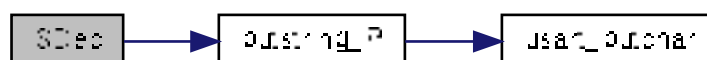
5.4.2.16 void SDec (void)

Definition at line 32 of file messages.c.

```

33 {
34     putstring_P( PSTR( "SDec\r" ) );
35 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



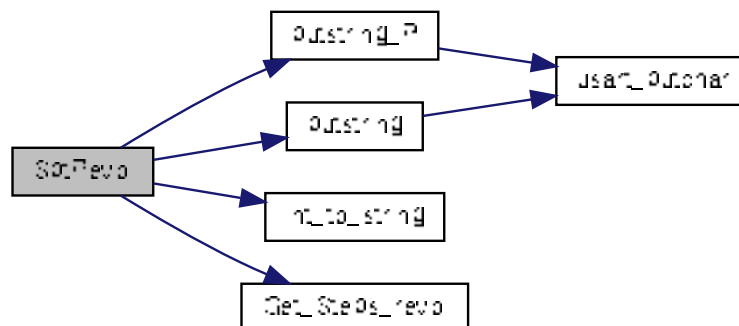
5.4.2.17 void SptRevol (void)

Definition at line 103 of file messages.c.

```

104 {
105     putstring_P( PSTR( "SPR=" ) );
106     putstring( int_to_string( ( uint16_t ) (
107         Get_Steps_revol( ) ) ) );
107     putstring_P( PSTR( "\r" ) );
108 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:

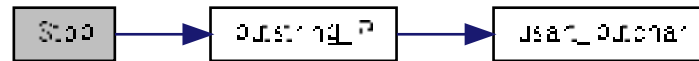


5.4.2.18 void Stop (void)

Definition at line 131 of file messages.c.

```
132 {  
133     putstring_P( PSTR( "Stop" ) );  
134 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.5 UART Library

Interrupt UART library using the built-in UART with transmit and receive circular buffers.

Macros

- `#define USART_BAUD_SELECT(baudRate, xtalCpu) (((xtalCpu)/((baudRate)*16l))-1)`
USART Baudrate Expression.
- `#define USART_BAUD_SELECT_DOUBLE_SPEED(baudRate, xtalCpu) (((xtalCpu)/((baudRate)*8l))-1)|0x8000)`
USART Baudrate Expression for ATmega double speed mode.
- `#define VEOL (char) 0x0d`

Functions

- `int usart_getchar (FILE *stream)`
Get received byte from ringbuffer.
- `void usart_init (unsigned int)`
Initialize USART and set baudrate.
- `int usart_putchar (char c, FILE *stream)`
Put byte to ringbuffer for transmitting via UART.

5.5.1 Detailed Description

Interrupt UART library using the built-in UART with transmit and receive circular buffers.

```
#include <usart.h>
```

This library can be used to transmit and receive data through the built in UART.

An interrupt is generated when the UART has finished transmitting or receiving a byte. The interrupt handling routines use circular buffers for buffering received and transmitted data.

The `USART_RX_BUFFER_SIZE` and `USART_TX_BUFFER_SIZE` constants define the size of the circular buffers in bytes. Note that these constants must be a power of 2. You may need to adapt this constants to your target and your application by adding `CDEFS += -DUART_RX_BUFFER_SIZE=nn -DUART_TX_BUFFER_SIZE=nn` to your Makefile.

Note

Based on Atmel Application Note AVR306

Author

Peter Fleury pfleury@gmx.ch <http://jump.to/fleury>

5.5.2 Macro Definition Documentation

5.5.2.1 `#define USART_BAUD_SELECT(baudRate, xtalCpu) (((xtalCpu)/((baudRate)*16l))-1)`

USART Baudrate Expression.

Parameters

<i>xtalcpu</i>	system clock in Mhz, e.g. 4000000L for 4Mhz
<i>baudrate</i>	baudrate in bps, e.g. 1200, 2400, 9600

Definition at line 87 of file usart.h.

5.5.2.2 `#define USART_BAUD_SELECT_DOUBLE_SPEED(baudRate, xtalCpu) (((xtalCpu)/((baudRate)*8L)-1)|0x8000)`

USART Baudrate Expression for ATmega double speed mode.

Parameters

<i>xtalcpu</i>	system clock in Mhz, e.g. 4000000L for 4Mhz
<i>baudrate</i>	baudrate in bps, e.g. 1200, 2400, 9600

Definition at line 93 of file usart.h.

5.5.2.3 `#define VEOL (char) 0x0d`

End of line character.

Definition at line 98 of file usart.h.

5.5.3 Function Documentation

5.5.3.1 `int usart_getchar (FILE * stream)`

Get received byte from ringbuffer.

Returns in the lower byte the received character and in the higher byte the last receive error. `UART_NO_DATA` is returned when no data is available.

Parameters

<i>void</i>	
-------------	--

Returns

lower byte: received byte from ringbuffer

higher byte: last receive status

- **0** successfully received data from UART

- **UART_NO_DATA**

no receive data available

- **UART_BUFFER_OVERFLOW**

Receive ringbuffer overflow. We are not reading the receive buffer fast enough, one or more received character have been dropped

- **UART_OVERRUN_ERROR**

Overrun condition by UART. A character already present in the UART UDR register was not read by the interrupt handler before the next character arrived, one or more received characters have been dropped.

- **UART_FRAME_ERROR**

Framing Error by UART

Read byte from ring buffer

Parameters

<i>FILE</i>	stream
-------------	--------

Returns

byte from ring buffer

Definition at line 128 of file usart.c.

```

129 {
130     uint8_t c;
131
132     if (usart_rx.fillcount == 0)
133         return EOF;
134
135     if (usart_rx.nlines == 0)
136         return EOF;
137
138     c = usart_rx.data[usart_rx.tail];
139     usart_rx.tail = (usart_rx.tail + 1) & (usart_rx.
size - 1);
140
141     //Disable RXC interrupt to run next lines atomic
142     UCSR0B &= ~(1 << RXCIE0);
143     usart_rx.fillcount--;
144
145     //Behavior similar to Unix stty ICRNL
146     if (c == VEOL)
147     {
148         c = '\n';
149         usart_rx.nlines--;
150     }
151
152     //Enable RXC interrupt
153     UCSR0B |= (1 << RXCIE0);
154
155     return c;
156 }/* usart_getchar */

```

5.5.3.2 void usart_init (unsigned int *baudrate*)

Initialize USART and set baudrate.

Parameters

<i>baudrate</i>	Specify baudrate using macro UART_BAUD_SELECT()
-----------------	---

Returns

none

Initialize USART and set baudrate Baudrate using macro [USART_BAUD_SELECT\(\)](#)

Parameters

<i>baudrate</i>	
-----------------	--

Definition at line 69 of file usart.c.

```

70 {
71     ATOMIC_BLOCK(ATOMIC_RESTORESTATE)
72     {
73         usart_rx.data = brx;
74         usart_rx.size = BSIZE;
75         usart_rx.head = 0;
76         usart_rx.tail = 0;
77         usart_rx.fillcount = 0;
78         usart_rx.nlines = 0;
79
80         usart_tx.data = btx;
81         usart_tx.size = BSIZE;

```

```

82     usart_tx.head = 0;
83     usart_tx.tail = 0;
84     usart_tx.fillcount = 0;
85     usart_tx.nlines = 0;
86
87     /* Set baud rate */
88     if (baudrate & 0x8000)
89     {
90         UCSR0A = (1 << U2X0); //Enable 2x speed
91         baudrate &= ~0x8000;
92     }
93     UBRR0H = (unsigned char) (baudrate >> 8);
94     UBRR0L = (unsigned char) baudrate;
95
96 #if (USART0_TYPE) == 485
97
98 #if defined(__AVR_ATmega8__) || defined(__AVR_ATmega16__) || defined(__AVR_ATmega32__) \
99     || defined(__AVR_ATmega323__)
100     PORTD |= (1 << PD0); //pull-up RX0
101 #elif defined(__AVR_ATmega162__)
102     PORTB |= (1 << PB2); //pull-up RX1
103 #else
104 #error "RX port is not internally pulled up"
105 #endif
106
107     //Receiver output enable
108     transmitter0_disable();
109     set_as_output(DDRB, transceiver0);
110
111     //Enable USART0 receiver and transmitter, receive complete
112     //and transmit complete interrupt.
113     UCSR0B = (1 << RXCIE0) | (1 << TXCIE0) | (1 << RXEN0) | (1 <<
114     TXEN0);
115 #else
116
117     /* Enable USART receiver and transmitter and receive complete interrupt */
118     UCSR0B = (1 << RXCIE0) | (1 << RXEN0) | (1 << TXEN0);
119 #endif
120
121     /* Set frame format: asynchronous, 8data, no parity, 1stop bit */
122     UCSR0C = (1 << URSEL0) | (1 << UCSZ01) | (1 << UCSZ00);
123 }
124 } /* usart_init */

```

Here is the caller graph for this function:



5.5.3.3 int usart_putchar (char c, FILE * stream)

Put byte to ringbuffer for transmitting via UART.

Parameters

<i>data</i>	byte to be transmitted
-------------	------------------------

Returns

none

Write byte to ring buffer for transmitting via USART

Parameters

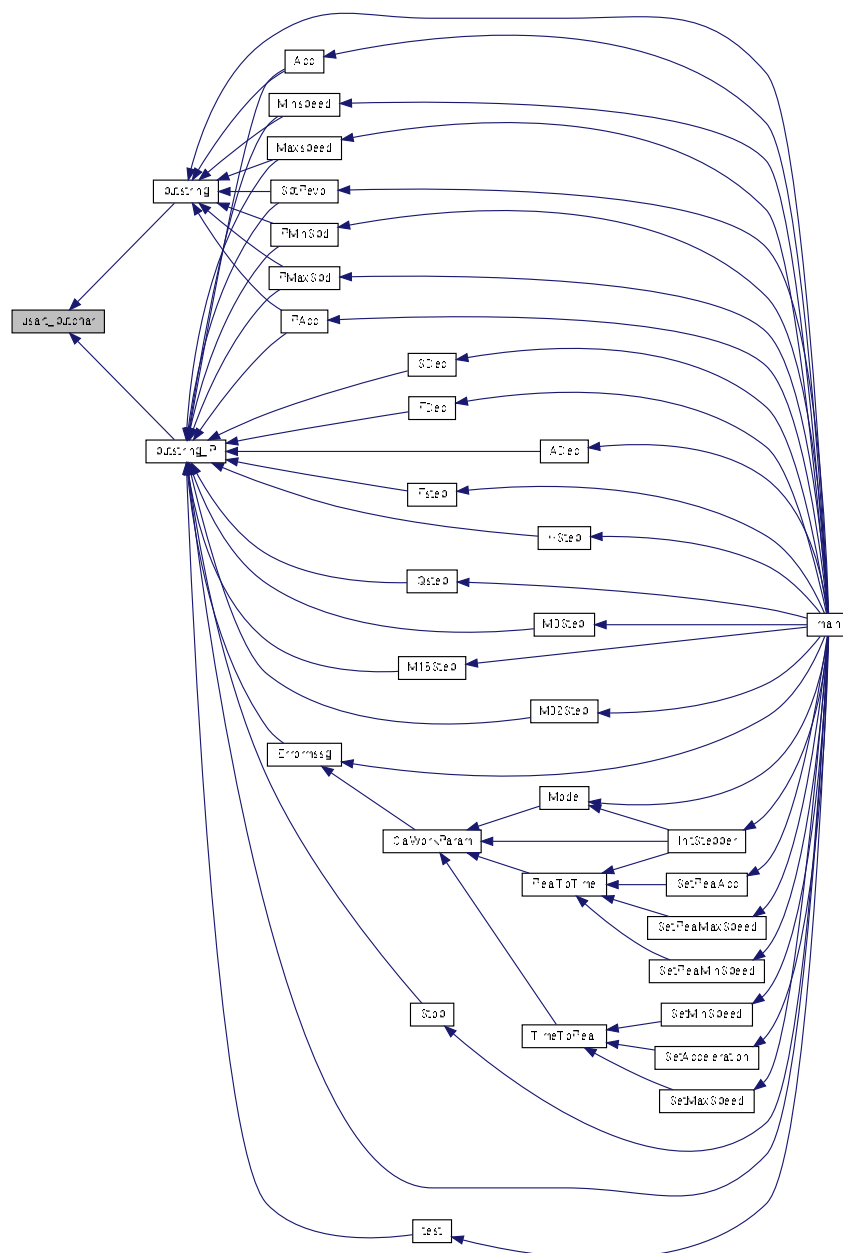
<i>c</i>	byte to be transmitted
<i>stream</i>	

Returns

Definition at line 164 of file usart.c.

```
165 {
166     #if (USART0_TYPE) == 485
167         //Driver output enable
168         transmitter0_enable();
169     #endif
170
171     if (c == '\n')
172         c = VEOL;
173
174     while (usart_tx.fillcount == usart_tx.size)
175         ; //wait for free space in buffer
176
177     usart_tx.data[usart_tx.head] = c;
178     usart_tx.head = (usart_tx.head + 1) & (usart_tx.size - 1);
179
180     //Disable UDRE interrupt to run next lines atomic
181     UCSROB &= ~(1 << UDRIE0);
182
183     usart_tx.fillcount++;
184
185     //Enable UDRE interrupt
186     UCSROB |= (1 << UDRIE0);
187
188     return 0;
189 } /* usart_putchar */
```

Here is the caller graph for this function:



5.6 utils Library

This module contains functions that are converting ASCII to uint64_t.

Functions

- uint64_t [ConvertASCItouint64](#) (char *in)
- char * [int_to_string](#) (uint64_t i)
- void [putstring](#) (const char *putc)
- void [putstring_P](#) (const char *putc)

5.6.1 Detailed Description

This module contains functions that are converting ASCII to uint64_t.

```
#include <utils.h>
```

This module contains functions that are converting ASCII to uint64_t

Note

Typical functions that are saving me memory.

Author

Bilyana Borisova bibishte@gmail.com

5.6.2 Function Documentation

5.6.2.1 uint64_t ConvertASCItouint64 (char * in)

Function that converts ASCII to uint64_t

Parameters

<i>in</i>	the first argument.
-----------	---------------------

Returns

uint64_t

Definition at line 45 of file utils.c.

```

46 {
47     char *b;
48     uint64_t c=0;
49     uint64_t multi=1;
50
51     b = in;
52
53     while ( ( (*b > 47) && (*b < 58 )) || ( *b == 32 ) || ( *b == 46 ) || ( *b ==44 ) )
54     {
55         b++;
56     }
57
58     while(b>=in)
59     {
60         if(( (*b > 47) && (*b < 58 )))
61         {
62             c=c+multi*(*b-'0');
63             multi=multi*10;
64         }

```

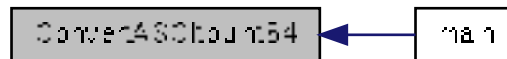


```

65
66     b--;
67 }
68 return c;
69 }

```

Here is the caller graph for this function:



5.6.2.2 char* int_to_string (uint64_t i)

Function that converts integer to string.

Parameters

<i>t</i>	the first argument.
----------	---------------------

Returns

char

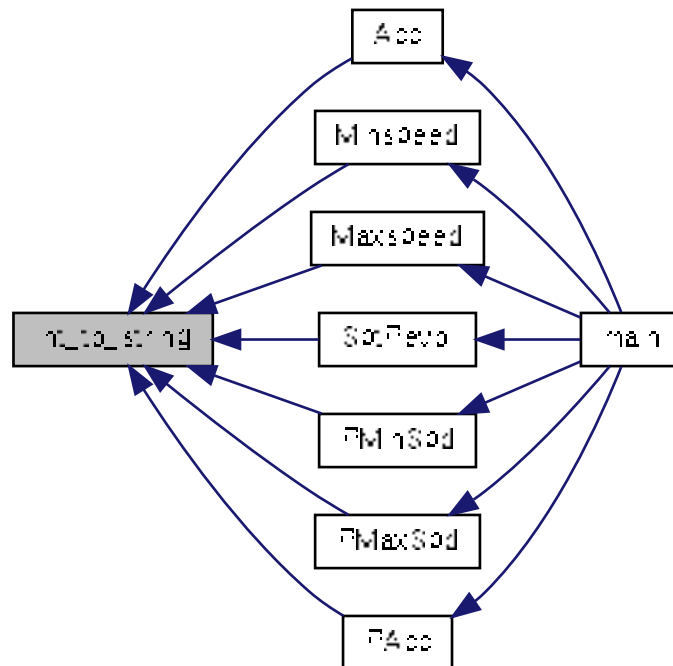
Definition at line 76 of file utils.c.

```

77 {
78     unsigned char temp;
79     unsigned char s = 0, t = 0;
80     if (i==0)
81     {
82         buf[0]='0';
83         buf[1]=0;
84         return buf;
85     }
86     while(i) {
87         buf[s++] = i % 10 + '0';
88         i /= (uint64_t)(10);
89     }
90     buf[s] = 0;
91     s--;
92     for (; t < s; t++, s--) {
93         temp = buf[s];
94         buf[s] = buf[t];
95         buf[t] = temp;
96     }
97     return buf;
98 }

```

Here is the caller graph for this function:



5.6.2.3 void putstring (const char * *putc*)

Function that replaces the printf for uint64_t.

Parameters

<i>putc</i>	the first argument.
-------------	---------------------

Returns

void

Definition at line 105 of file utils.c.

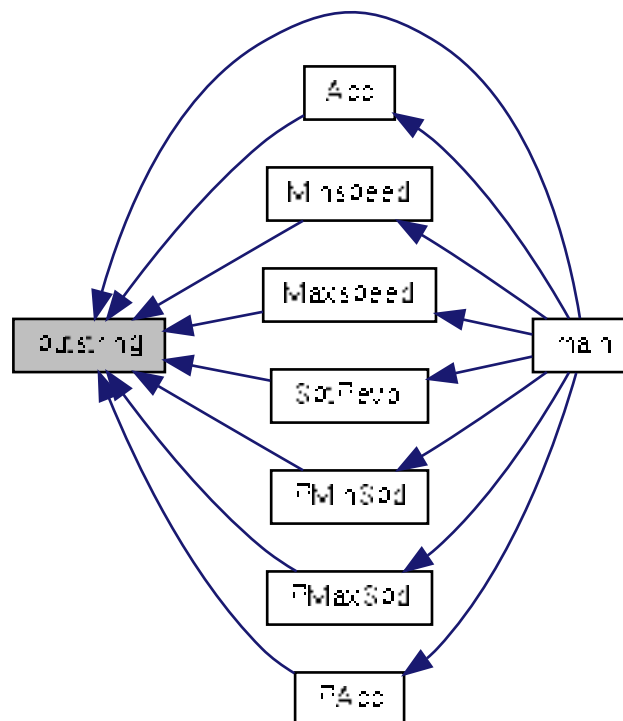
```

106 {
107     const char *p;
108     p=putc;
109     while (*p!=0)
110     {
111         usart_putchar(*p, stdout);
112         p++;
113     }
114
115     return;
116 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.2.4 void putstring_P (const char * *putc*)

Function that replaces the printf from Programme memory space.

Parameters

<i>putc</i>	the first argument.
-------------	---------------------

Returns

void

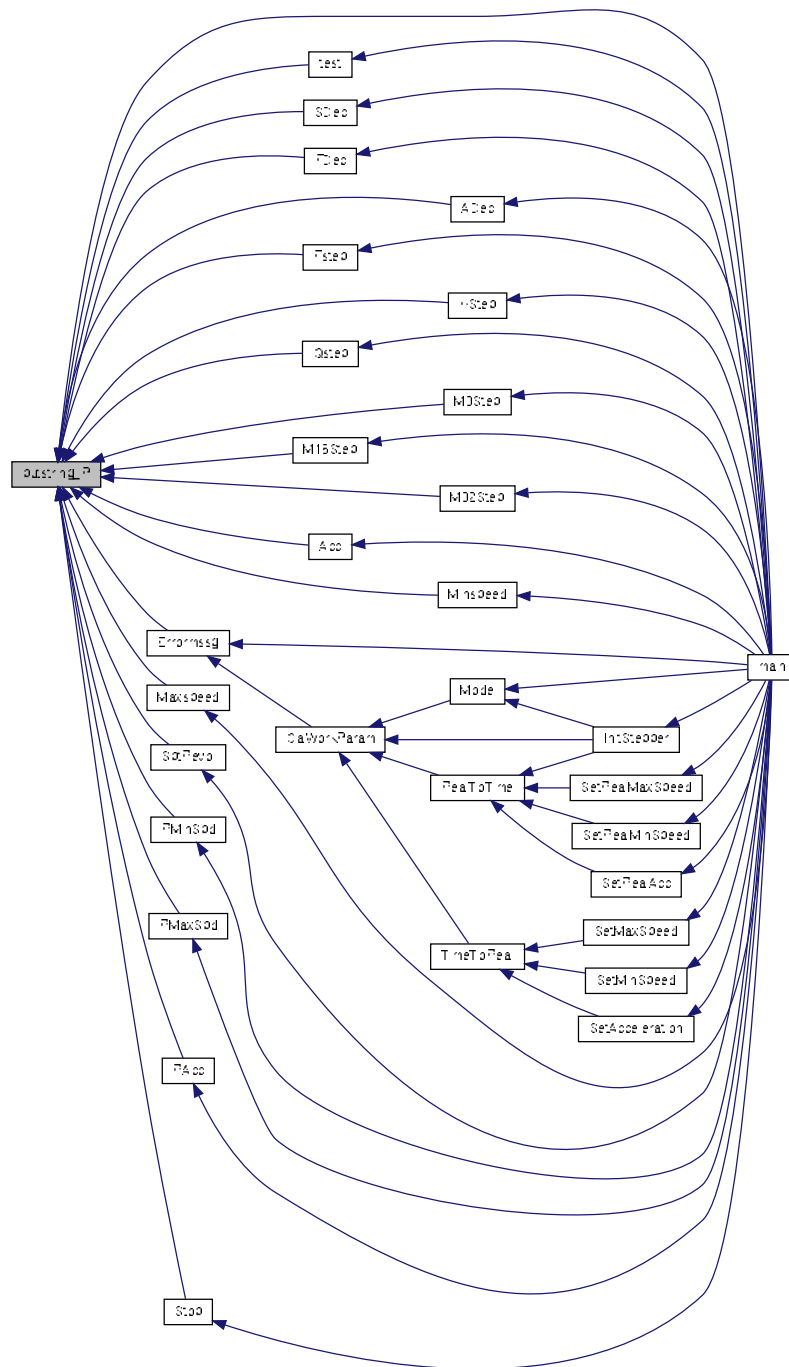
Definition at line 124 of file utils.c.

```
125 {  
126     while (pgm_read_byte ( putc )!=0)  
127     {  
128         usart_putchar(pgm_read_byte ( putc ), stdout);  
129         putc++;  
130     }  
131     return;  
132 }  
133 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

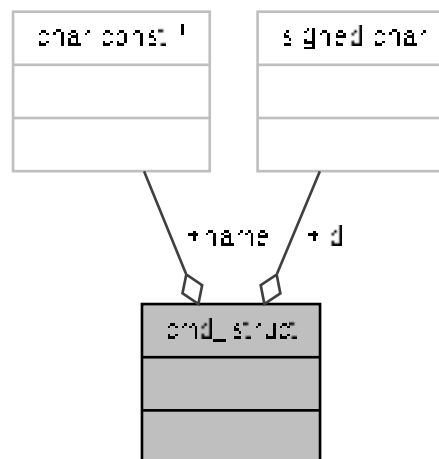


Chapter 6

Data Structure Documentation

6.1 cmd_struct Struct Reference

Collaboration diagram for cmd_struct:



Data Fields

- signed char `id`
- char const * `name`

6.1.1 Detailed Description

Definition at line 13 of file `interpreter.c`.

6.1.2 Field Documentation

6.1.2.1 signed char cmd_struct::id

Definition at line 15 of file interpreter.c.

6.1.2.2 char const* cmd_struct::name

Definition at line 16 of file interpreter.c.

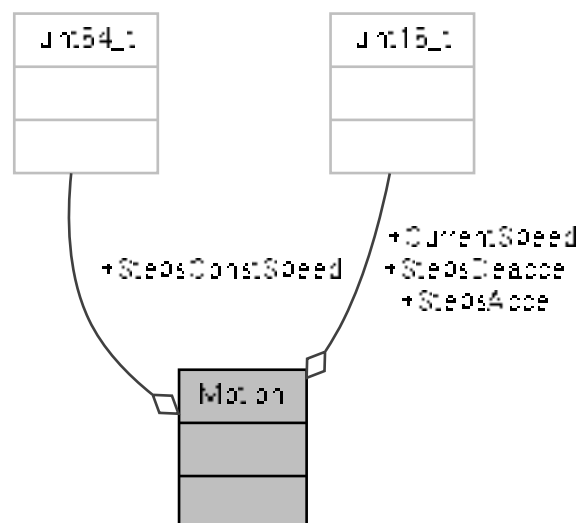
The documentation for this struct was generated from the following file:

- [interpreter.c](#)

6.2 Motion Struct Reference

This structure defines the current step execution status. This structure is used for the timer interrupting and in the comments for initializing the spin.

Collaboration diagram for Motion:



Data Fields

- uint16_t [CurrentSpeed](#)
- uint16_t [StepsAccel](#)
- uint64_t [StepsConstSpeed](#)
- uint16_t [StepsDeaccel](#)

6.2.1 Detailed Description

This structure defines the current step execution status. This structure is used for the timer interrupting and in the comments for initializing the spin.

Definition at line 73 of file drv_8825.c.

6.2.2 Field Documentation

6.2.2.1 Motion::CurrentSpeed

Member 'CurrentSpeed' defines current speed saved in the timer

Definition at line 78 of file drv_8825.c.

6.2.2.2 Motion::StepsAccel

Member 'StepsAccel' defines the remaining steps for acceleration process

Definition at line 75 of file drv_8825.c.

6.2.2.3 Motion::StepsConstSpeed

Member 'StepsConstSpeed' defines remaining steps for constant speed

Definition at line 76 of file drv_8825.c.

6.2.2.4 Motion::StepsDeaccel

Member 'StepsDeaccel' defines the remaining steps for deacceleration

Definition at line 77 of file drv_8825.c.

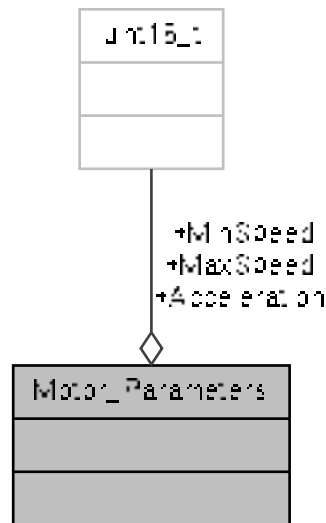
The documentation for this struct was generated from the following file:

- [drv_8825.c](#)

6.3 Motor_Parameters Struct Reference

This structure defines types for motor parameters. This type is used for declaration of three structures that are for the motor parameters used for timer, the real parameters of the motor and the working parameters that have been already calculated according to the microstep mode.

Collaboration diagram for Motor_Parameters:



Data Fields

- uint16_t [Acceleration](#)
- uint16_t [MaxSpeed](#)
- uint16_t [MinSpeed](#)

6.3.1 Detailed Description

This structure defines types for motor parameters This type is used for declaration of three structures that are for the motor parameters used for timer, the real parameters of the motor and the working parameters that have been already calculated according to the microstep mode.

Definition at line 52 of file drv_8825.c.

6.3.2 Field Documentation

6.3.2.1 Motor_Parameters::Acceleration

Member 'Acceleration' defines the step of acceleration

Definition at line 56 of file drv_8825.c.

6.3.2.2 Motor_Parameters::MaxSpeed

Member 'MaxSpeed' defines the maximum allowed speed

Definition at line 54 of file drv_8825.c.

6.3.2.3 Motor_Parameters::MinSpeed

Member 'MinSpeed' defines the minimum allowed speed

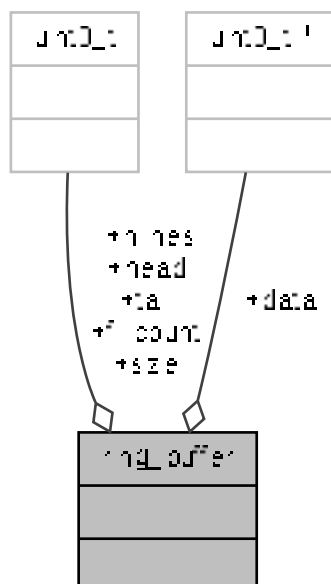
Definition at line 55 of file drv_8825.c.

The documentation for this struct was generated from the following file:

- [drv_8825.c](#)

6.4 ring_buffer Struct Reference

Collaboration diagram for ring_buffer:



Data Fields

- `uint8_t * data`
- `uint8_t fillcount`
- `uint8_t head`
- `uint8_t nlines`
- `uint8_t size`
- `uint8_t tail`

6.4.1 Detailed Description

Definition at line 39 of file usart.c.

6.4.2 Field Documentation

6.4.2.1 `uint8_t* ring_buffer::data`

Definition at line 41 of file `usart.c`.

6.4.2.2 `uint8_t ring_buffer::fillcount`

Definition at line 45 of file `usart.c`.

6.4.2.3 `uint8_t ring_buffer::head`

Definition at line 43 of file `usart.c`.

6.4.2.4 `uint8_t ring_buffer::nlines`

Definition at line 46 of file `usart.c`.

6.4.2.5 `uint8_t ring_buffer::size`

Definition at line 42 of file `usart.c`.

6.4.2.6 `uint8_t ring_buffer::tail`

Definition at line 44 of file `usart.c`.

The documentation for this struct was generated from the following file:

- [usart.c](#)

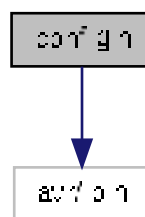
Chapter 7

File Documentation

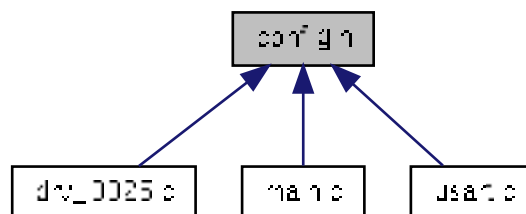
7.1 config.h File Reference

```
#include <avr/io.h>
```

Include dependency graph for config.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define [check_pin](#)(PINx, PINxn) (PINx & (1 << PINxn))

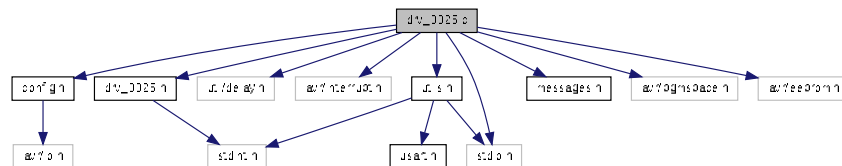
- #define CHECKPIN(x, y) ((x&(1<<y))!=0?1:0)
- #define clr_bit(byte, bit) byte &= ~(1 << bit)
- #define clr_port(PORTx, PORTxn) PORTx &= ~(1 << PORTxn)
- #define DDECAY DDRC
- #define DDIR_485 DDRB
- #define DDIRECTION DDRB
- #define DECAY PC1
- #define DENABLE_STEPPER DDRD
- #define DFAULT DDRB
- #define DHOME DDRC
- #define DIR_485 PB0
- #define DIRECTION PB2
- #define DMISO DDRB
- #define DMODE0 DDRC
- #define DMODE1 DDRC
- #define DMODE2 DDRC
- #define DMOSI DDRB
- #define DRXD DDRD
- #define DSCL DDRB
- #define DSTEP DDRC
- #define DTXD DDRD
- #define ENABLE_STEPPER PD5
- #define false 1!=1
- #define FAULT PB1
- #define HOME PC2
- #define IDECAY PINC
- #define IDIR_485 PINB
- #define IDIRECTION PINB
- #define IENABLE_STEPPER PIND
- #define IFAULT PINB
- #define IHOME PINC
- #define IMISO PINB
- #define IMODE0 PINC
- #define IMODE1 PINC
- #define IMODE2 PINC
- #define IMOSI PINB
- #define IRXD PIND
- #define ISCL PINB
- #define ISTEP PINC
- #define ITXD PIND
- #define MISO PB4
- #define MODE0 PC3
- #define MODE1 PC4
- #define MODE2 PC5
- #define MOSI PB3
- #define PDECAY PORTC
- #define PDIR_485 PORTB
- #define PDIRECTION PORTB
- #define PENABLE_STEPPER PORTD
- #define PFAULT PORTB
- #define PHOME PORTC
- #define PMISO PORTB
- #define PMODE0 PORTC
- #define PMODE1 PORTC
- #define PMODE2 PORTC

- #define [PMOSI](#) PORTB
- #define [PRXD](#) PORTD
- #define [PSCL](#) PORTB
- #define [PSTEP](#) PORTC
- #define [PTXD](#) PORTD
- #define [RXD](#) PD0
- #define [SCL](#) PB5
- #define [set_as_input](#)(DDRx, DDxn) DDRx &= ~(1 << DDxn)
- #define [set_as_output](#)(DDRx, DDxn) DDRx |= (1 << DDxn)
- #define [set_bit](#)(byte, bit) byte |= (1 << bit)
- #define [set_port](#)(PORTx, PORTxn) PORTx |= (1 << PORTxn)
- #define [STEP](#) PC0
- #define [true](#) 1==1
- #define [TXD](#) PD1

7.2 drv_8825.c File Reference

```
#include "drv_8825.h"
#include "config.h"
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include "utils.h"
#include "messages.h"
#include <avr/pgmspace.h>
#include <avr/eeprom.h>
```

Include dependency graph for drv_8825.c:



Data Structures

- struct [Motion](#)

This structure defines the current step execution status. This structure is used for the timer interrupting and in the comments for initializing the spin.

- struct [Motor_Parameters](#)

This structure defines types for motor parameters. This type is used for declaration of three structures that are for the motor parameters used for timer, the real parameters of the motor and the working parameters that have been already calculated according to the microstep mode.

Macros

- #define [PERTMR](#) 723
- #define [TIMERCONST](#) 82987552

Typedefs

- typedef struct [Motor_Parameters](#) [Motor_Parameters](#)

Functions

- void [CalWorkParam](#) (void)
- void [Count_Step](#) ([step_type](#) step, uint64_t step_count)
- void [Decay](#) ([decay_type](#) decay)
- void [Disabled_Stepper](#) (void)
- void [Enable_Stepper](#) (void)
- [decay_type](#) [Get_Decay](#) (void)
- [mode_type](#) [Get_Mode](#) (void)
- uint8_t [Get_Start](#) (void)
- uint16_t [Get_Steps_revol](#) (void)
- uint16_t [GetAcceleration](#) (void)
- uint16_t [GetCurrentSpeed](#) (void)
- uint16_t [GetMaxSpeed](#) (void)
- uint16_t [GetMinSpeed](#) (void)
- uint16_t [GetRealAcc](#) (void)
- uint16_t [GetRealMaxSpeed](#) (void)
- uint16_t [GetRealMinSpeed](#) (void)
- uint64_t [HelperRtoT](#) (uint64_t in)
- uint64_t [HelperTtoR](#) (uint64_t in)
- uint8_t [Home](#) (void)
- void [InitStepper](#) (void)
- [ISR](#) (TIMER1_OVF_vect)
- void [Mode](#) ([mode_type](#) mode)
- void [RealToTime](#) (void)
- void [Set_Steps_revol](#) (uint16_t step_rev)
- void [SetAcceleration](#) (uint16_t accel)
- void [SetCurrentSpeed](#) (uint16_t spd)
- void [SetMaxSpeed](#) (uint16_t spd)
- void [SetMinSpeed](#) (uint16_t spd)
- void [SetRealAcc](#) (uint16_t realacc)
- void [SetRealMaxSpeed](#) (uint16_t maxspd)
- void [SetRealMinSpeed](#) (uint16_t minspd)
- void [Step](#) ([step_type](#) step)
- void [Stop_Motion_fast](#) (void)
- void [Stop_Motion_normal](#) (void)
- void [store](#) (void)
- void [TimeToReal](#) (void)
- void [Way_Speed](#) ([step_type](#) step)

Variables

- static volatile uint8_t [ConstSpd](#)
- static volatile [decay_type](#) [CurrentDecay](#)
- static volatile [mode_type](#) [CurrentMode](#)
- static volatile uint8_t [ModeStart](#)
- static volatile struct [Motion](#) [Motion](#)
- static volatile [Motor_Parameters](#) [RealSpeed](#)
- static volatile [step_type](#) [StepDir](#)
- static volatile uint16_t [StepsPerRev](#)
- static volatile [Motor_Parameters](#) [TimeParam](#)
- static volatile [Motor_Parameters](#) [WorkingParam](#)

7.2.1 Macro Definition Documentation

7.2.1.1 #define PERTMR 723

Defines the Timer

Definition at line 37 of file drv_8825.c.

7.2.1.2 #define TIMERCONST 82987552

Calculate the Timer Constant

Definition at line 38 of file drv_8825.c.

7.2.2 Typedef Documentation

7.2.2.1 typedef struct Motor_Parameters Motor_Parameters

7.2.3 Function Documentation

7.2.3.1 ISR (TIMER1_OVF_vect)

Interrupt sours routine.Makes a step and then loads the next value of the timer

Definition at line 477 of file drv_8825.c.

```

478 {
479     if ( ConstSpd == 1 )
480     {
481         Motion.StepsConstSpeed++;
482     }
483     if ( Motion.StepsAccel > 0 )
484     {
485         if ( Motion.CurrentSpeed > WorkingParam.
Acceleration )
486         {
487             Motion.CurrentSpeed = Motion.CurrentSpeed -
WorkingParam.Acceleration;
488         } else
489         {
490             Motion.CurrentSpeed = WorkingParam.
MaxSpeed;
491         }
492     }
493     if ( Motion.CurrentSpeed < WorkingParam.
MaxSpeed )
494     {
495         Motion.CurrentSpeed = WorkingParam.
MaxSpeed;
496     }
497     TCNT1 = 0 - Motion.CurrentSpeed;
498     Step( StepDir );
499     Motion.StepsAccel--;
500
501     return;
502 }
503
504 if ( Motion.StepsConstSpeed > 0 )
505 {
506     Motion.StepsConstSpeed--;
507     TCNT1 = 0 - WorkingParam.MaxSpeed;
508     Step( StepDir );
509
510     return;
511 }
512
513
514 if ( Motion.StepsDeaccel > 0 )
515 {
516     Motion.StepsDeaccel--;
517     Motion.CurrentSpeed = Motion.CurrentSpeed +
WorkingParam.Acceleration;
518     TCNT1 = 0 - Motion.CurrentSpeed;

```

```

519         Step( StepDir );
520
521         return;
522     }
523
524     clr_bit( TIMSK, TOIE1 );
525     TCNT1 = 65535 - WorkingParam.MinSpeed;
526 }

```

Here is the call graph for this function:



7.2.4 Variable Documentation

7.2.4.1 volatile uint8_t ConstSpd [static]

Flag used for constant speed mode.

Definition at line 97 of file drv_8825.c.

7.2.4.2 volatile decay_type CurrentDecay [static]

Store the current decay mode.

Definition at line 102 of file drv_8825.c.

7.2.4.3 volatile mode_type CurrentMode [static]

Store the current step mode.

Definition at line 106 of file drv_8825.c.

7.2.4.4 volatile uint8_t ModeStart [static]

Flag for enable and disable stepper motor

Definition at line 110 of file drv_8825.c.

7.2.4.5 volatile struct Motion Motion [static]

7.2.4.6 volatile Motor_Parameters RealSpeed [static]

This is used for calculating the timer values. This is the real speed of the motor.

Definition at line 93 of file drv_8825.c.

7.2.4.7 volatile step_type StepDir [static]

Flag for stepper direction.

Definition at line 114 of file drv_8825.c.

7.2.4.8 volatile uint16_t StepsPerRev [static]

Motor parameter - Steps per revolution.

Definition at line 119 of file drv_8825.c.

7.2.4.9 volatile Motor_Parameters TimeParam [static]

This is used for calculating the timer values for full step mode.

Definition at line 85 of file drv_8825.c.

7.2.4.10 volatile Motor_Parameters WorkingParam [static]

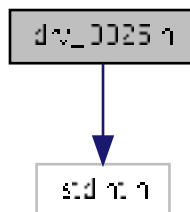
This is used for the timer and it is calculated based on the mode.

Definition at line 89 of file drv_8825.c.

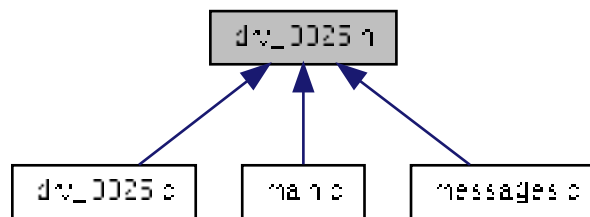
7.3 drv_8825.h File Reference

```
#include <stdint.h>
```

Include dependency graph for drv_8825.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum [decay_type](#) { [SLOW_DECAY](#), [FAST_DECAY](#), [AUTO_DECAY](#) }
Type for decay status.
- enum [mode_type](#) { [MODE_FULL_STEP](#), [MODE_HALF_STEP](#), [MODE_QUATER_STEP](#), [MODE_8_MICROSTEP](#), [MODE_16_MICROSTEP](#), [MODE_32_MICROSTEP](#) }
Type for mode status.
- enum [step_type](#) { [STEP_COUNTER_CLOCKWISE](#), [STEP_CLOCKWISE](#) }
Type for the direction status.

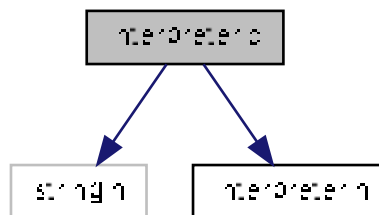
Functions

- void [CalWorkParam](#) (void)
- void [Count_Step](#) ([step_type](#) step, uint64_t step_count)
- void [Decay](#) ([decay_type](#) decay)
- void [Disabled_Stepper](#) (void)
- void [Enable_Stepper](#) (void)
- [decay_type](#) [Get_Decay](#) (void)
- [mode_type](#) [Get_Mode](#) (void)
- uint8_t [Get_Start](#) (void)
- uint16_t [Get_Steps_revol](#) (void)
- uint16_t [GetAcceleration](#) (void)
- uint16_t [GetCurrentSpeed](#) (void)
- uint16_t [GetMaxSpeed](#) (void)
- uint16_t [GetMinSpeed](#) (void)
- uint16_t [GetRealAcc](#) (void)
- uint16_t [GetRealMaxSpeed](#) (void)
- uint16_t [GetRealMinSpeed](#) (void)
- uint64_t [HelperRtoT](#) (uint64_t)
- uint64_t [HelperTtoR](#) (uint64_t)
- uint8_t [Home](#) (void)
- void [InitStepper](#) (void)
- void [Mode](#) ([mode_type](#) mode)
- void [RealToTime](#) (void)
- void [Set_Steps_revol](#) (uint16_t step_rev)
- void [SetAcceleration](#) (uint16_t accel)

- void [SetCurrentSpeed](#) (uint16_t spd)
- void [SetMaxSpeed](#) (uint16_t spd)
- void [SetMinSpeed](#) (uint16_t spd)
- void [SetRealAcc](#) (uint16_t realacc)
- void [SetRealMaxSpeed](#) (uint16_t maxspd)
- void [SetRealMinSpeed](#) (uint16_t minspd)
- void [Step](#) (step_type step)
- void [Stop_Motion_fast](#) (void)
- void [Stop_Motion_normal](#) (void)
- void [store](#) (void)
- void [TimeToReal](#) (void)
- void [Way_Speed](#) (step_type step)

7.4 interpreter.c File Reference

```
#include <string.h>
#include "interpreter.h"
Include dependency graph for interpreter.c:
```



Data Structures

- struct [cmd_struct](#)

Functions

- char [get_cmd_id](#) (char *name)
- char const * [get_cmd_name](#) (char id)

Variables

- struct [cmd_struct](#) [cmd_tbl](#) []

7.4.1 Function Documentation

7.4.1.1 char [get_cmd_id](#) (char * *name*)

Parameters

<i>name</i>	
-------------	--

Returns

Definition at line 60 of file interpreter.c.

```

61 {
62     unsigned char i = 0;
63
64     if (name[0] == 0)
65         return -2;
66
67     for (i = 0; cmd_tbl[i].id != -1; i++)
68         if (!strcmp(name, cmd_tbl[i].name, strlen(cmd_tbl[i].name)))
69             return cmd_tbl[i].id;
70
71     return -1;
72 }
```

Here is the caller graph for this function:



7.4.1.2 char const* get_cmd_name (char id)

Parameters

<i>id</i>	
-----------	--

Returns

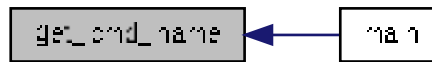
command name or NULL if not found

Definition at line 79 of file interpreter.c.

```

80 {
81     unsigned char i;
82
83     for (i = 0; cmd_tbl[i].id != -1; i++)
84         if (id == cmd_tbl[i].id)
85             return cmd_tbl[i].name;
86
87     return NULL;
88 }
```

Here is the caller graph for this function:

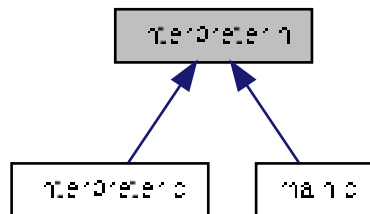


7.4.2 Variable Documentation

7.4.2.1 struct cmd_struct cmd_tbl[]

7.5 interpreter.h File Reference

This graph shows which files directly or indirectly include this file:



Enumerations

- enum {
[VERSION](#), [HELP](#), [TEST](#), [SET_DEC_SLOW](#),
[SET_DEC_FAST](#), [SET_DEC_AUTO](#), [SET_MODE_FULL](#), [SET_MODE_HALF](#),
[SET_MODE_QUATER](#), [SET_MODE_8](#), [SET_MODE_16](#), [SET_MODE_32](#),
[SET_HOME](#), [SET_ACC](#), [SET_MIN_SPEED](#), [SET_MAX_SPEED](#),
[SET_STP_REV](#), [SET_R_MIN_SPD](#), [SET_R_MAX_SPD](#), [SET_R_ACC](#),
[GET_ACC](#), [GET_MIN_SPEED](#), [GET_MAX_SPEED](#), [GET_STP_REV](#),
[GET_R_MIN_SPD](#), [GET_R_MAX_SPD](#), [GET_R_ACC](#), [ENABLE](#),
[DISABLE](#), [GOTO](#), [STOPN](#), [STOPF](#),
[CONSTSPD](#), [STATUS](#) }

Functions

- char [get_cmd_id](#) (char *)
- char const * [get_cmd_name](#) (char)

7.5.1 Enumeration Type Documentation

7.5.1.1 anonymous enum

Enumerator

VERSION
HELP
TEST
SET_DEC_SLOW
SET_DEC_FAST
SET_DEC_AUTO
SET_MODE_FULL
SET_MODE_HALF
SET_MODE_QUATER
SET_MODE_8
SET_MODE_16
SET_MODE_32
SET_HOME
SET_ACC
SET_MIN_SPEED
SET_MAX_SPEED
SET_STP_REV
SET_R_MIN_SPD
SET_R_MAX_SPD
SET_R_ACC
GET_ACC
GET_MIN_SPEED
GET_MAX_SPEED
GET_STP_REV
GET_R_MIN_SPD
GET_R_MAX_SPD
GET_R_ACC
ENABLE
DISABLE
GOTO
STOPN
STOPF
CONSTSPD
STATUS

Definition at line 11 of file interpreter.h.

```
11     {  
12         VERSION,  
13         HELP,  
14         TEST,  
15         SET_DEC_SLOW,  
16         SET_DEC_FAST,  
17         SET_DEC_AUTO,  
18         SET_MODE_FULL,
```



```

19     SET_MODE_HALF,
20     SET_MODE_QUATER,
21     SET_MODE_8,
22     SET_MODE_16,
23     SET_MODE_32,
24     SET_HOME,
25     SET_ACC,
26     SET_MIN_SPEED,
27     SET_MAX_SPEED,
28     SET_STP_REV,
29     SET_R_MIN_SPD,
30     SET_R_MAX_SPD,
31     SET_R_ACC,
32     GET_ACC,
33     GET_MIN_SPEED,
34     GET_MAX_SPEED,
35     GET_STP_REV,
36     GET_R_MIN_SPD,
37     GET_R_MAX_SPD,
38     GET_R_ACC,
39     ENABLE,
40     DISABLE,
41     GOTO,
42     STOPN,
43     STOPF,
44     CONSTSPD,
45     STATUS
46
47 };

```

7.5.2 Function Documentation

7.5.2.1 char get_cmd_id (char * name)

Parameters

<i>name</i>	
-------------	--

Returns

Definition at line 60 of file interpreter.c.

```

61 {
62     unsigned char i = 0;
63
64     if (name[0] == 0)
65         return -2;
66
67     for (i = 0; cmd_tbl[i].id != -1; i++)
68         if (!strcmp(name, cmd_tbl[i].name, strlen(cmd_tbl[i].name)))
69             return cmd_tbl[i].id;
70
71     return -1;
72 }

```

Here is the caller graph for this function:



7.5.2.2 char const* get_cmd_name (char *id*)

Parameters

<i>id</i>	
-----------	--

Returns

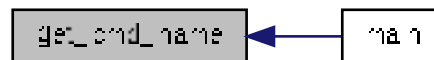
command name or NULL if not found

Definition at line 79 of file interpreter.c.

```

80 {
81     unsigned char i;
82
83     for (i = 0; cmd_tbl[i].id != -1; i++)
84         if (id == cmd_tbl[i].id)
85             return cmd_tbl[i].name;
86
87     return NULL;
88 }
```

Here is the caller graph for this function:



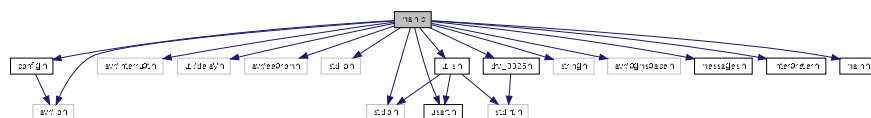
7.6 main.c File Reference

```

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <avr/eeprom.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <avr/pgmspace.h>
#include "usart.h"
#include "messages.h"
#include "interpreter.h"
#include "config.h"
#include "drv_8825.h"
#include "utils.h"
#include "main.h"

```

Include dependency graph for main.c:



Macros

- `#define F_CPU 11059200UL`
define CPU frequency in Mhz here if not defined in Makefile
- `#define USART_BAUD_RATE 115200`
115200 baud

Functions

- void `Init_Input_Output` (void)
- `int16_t main` (void)
main loop
- void `test` (void)

Variables

- FILE `usart_str`

7.6.1 Macro Definition Documentation

7.6.1.1 `#define F_CPU 11059200UL`

define CPU frequency in Mhz here if not defined in Makefile

Definition at line 46 of file main.c.

7.6.1.2 `#define USART_BAUD_RATE 115200`

115200 baud

Definition at line 50 of file main.c.

7.6.2 Function Documentation

7.6.2.1 `int16_t main (void)`

main loop

Definition at line 62 of file main.c.

```

63 {
64     char message[32]; // 1 byte + for string termination
65
66     int8_t cmd;
67     const char *cptr;
68     char * pch;
69     uint64_t ch;
70     uint8_t i;
71
72
73     SFIOR &= 0xfb; //11111011 -> PUD=0
74     /*
75      * Set ports data direction
76
77      */
78     Init_Input_Output ( );
79     usart_init( USART_BAUD_SELECT( USART_BAUD_RATE,
80                                   F_CPU ) );
81
82

```

```

83      /*
84      * now enable interrupt, since USART library is interrupt controlled

85      */
86      sei( );
87      stdout = stdin = &usart_str;
88
89      _delay_ms( 100 );
90
91
92      InitStepper();
93
94
95      for (;; )
96      {
97
98          if ( ( cptr = fgets( message, sizeof message - 1, stdin ) ) )
99          {
100
101
102              cmd = get_cmd_id( message );
103
104              switch ( cmd )
105              {
106                  case VERSION:
107                      putstring_P( PSTR( "1.0\r" ) );
108                      break;
109
110
111                  case HELP:
112                      // putstring_P( PSTR( "help?\r" ) );
113
114                      i = 0;
115
116                      do
117                      {
118                          cptr = get_cmd_name( i++ );
119                          if ( cptr )
120                          {
121                              putstring( cptr );
122                              putstring_P( PSTR( "\r" ) );
123                          }
124                      }
125                      while ( cptr );
126
127                      break;
128
129                  case TEST:
130                      test( );
131                      break;
132
133                  case SET_DEC_SLOW:
134                      Decay( SLOW_DECAY );
135                      SDec( );
136                      break;
137
138                  case SET_DEC_FAST:
139                      Decay( FAST_DECAY );
140                      FDec( );
141                      break;
142
143                  case SET_DEC_AUTO:
144                      Decay( AUTO_DECAY );
145                      ADec( );
146                      break;
147
148                  case SET_MODE_FULL:
149                      Mode( MODE_FULL_STEP );
150                      Fstep( );
151                      break;
152
153                  case SET_MODE_HALF:
154                      Mode( MODE_HALF_STEP );
155                      HStep( );
156                      break;
157
158                  case SET_MODE_QUATER:
159                      Mode( MODE_QUATER_STEP );
160                      Qstep( );
161                      break;
162
163                  case SET_MODE_8:
164                      Mode( MODE_8_MICROSTEP );
165                      M8Step( );
166                      break;
167
168                  case SET_MODE_16:

```

```

169         Mode( MODE_16_MICROSTEP );
170         M16Step( );
171         break;
172
173     case SET_MODE_32:
174         Mode( MODE_32_MICROSTEP );
175         M32Step( );
176         break;
177
178     case SET_HOME:
179         //          putstring("home%d\r", Home());
180         putstring_P( PSTR( "home " ) );
181         if ( !(check_pin( IHOME, HOME )) )
182         {
183             putstring_P( PSTR( "TRUE" ) );
184         } else
185         {
186             putstring_P( PSTR( "FALSE" ) );
187         }
188         //          putstring(ConvertASCItouint64(Home()));
189         putstring_P( PSTR( "\r" ) );
190
191         break;
192
193     case SET_ACC:
194         pch = strstr( message, "=" );
195         if ( pch != NULL )
196         {
197             pch = strstr( pch, "=" );
198             ch = ConvertASCItouint64( pch + 1 );
199             if ( ch > 0 )
200             {
201                 SetAcceleration( ch );
202                 Acc( );
203                 break;
204             } else
205             {
206                 Errormssg( );
207                 break;
208             }
209         }
210         break;
211
212     case SET_MIN_SPEED:
213         pch = strstr( message, "=" );
214         if ( pch != NULL )
215         {
216             pch = strstr( pch, "=" );
217             ch = ConvertASCItouint64( pch + 1 );
218             if ( ch > 0 )
219             {
220                 SetMinSpeed( ch );
221                 Minspeed( );
222                 break;
223             } else
224             {
225                 Errormssg( );
226                 break;
227             }
228         }
229         break;
230
231     case SET_MAX_SPEED:
232         pch = strstr( message, "=" );
233         if ( pch != NULL )
234         {
235             pch = strstr( pch, "=" );
236             ch = ConvertASCItouint64( pch + 1 );
237             if ( ch > 0 )
238             {
239                 SetMaxSpeed( ch );
240                 Maxspeed( );
241                 break;
242             } else
243             {
244                 Errormssg( );
245                 break;
246             }
247         }
248         break;
249
250     case SET_STP_REV:
251
252         pch = strstr( message, "=" );
253         if ( pch != NULL )
254         {
255             pch = strstr( pch, "=" );

```

```

256         ch = ConvertASCItouint64( pch + 1 );
257         if ( ch > 0 )
258         {
259             Set_Steps_revol( ch );
260             SptRevol();
261             break;
262         }
263         else
264         {
265             Errormssg( );
266             break;
267         }
268     }
269     break;
270
271 case SET_R_MIN_SPD:
272
273     pch = strstr( message, "=" );
274     if ( pch != NULL )
275     {
276         //pch = strstr( pch, "=" );
277         ch = ConvertASCItouint64( pch + 1 );
278         if ( ch > 0 )
279         {
280             SetRealMinSpeed( ch );
281             RMinSpd();
282             break;
283         }
284         else
285         {
286             Errormssg( );
287             break;
288         }
289     }
290
291     break;
292
293 case SET_R_MAX_SPD:
294
295     pch = strstr( message, "=" );
296     if ( pch != NULL )
297     {
298         pch = strstr( pch, "=" );
299         ch = ConvertASCItouint64( pch + 1 );
300         if ( ch > 0 )
301         {
302             SetRealMaxSpeed( ch );
303             RMaxSpd();
304             break;
305         }
306         else
307         {
308             Errormssg( );
309             break;
310         }
311     }
312
313     break;
314
315 case SET_R_ACC:
316
317     pch = strstr( message, "=" );
318     if ( pch != NULL )
319     {
320         pch = strstr( pch, "=" );
321         ch = ConvertASCItouint64( pch + 1 );
322         if ( ch > 0 )
323         {
324             SetRealAcc( ch );
325             RAcc();
326             break;
327         }
328         else
329         {
330             Errormssg( );
331             break;
332         }
333     }
334
335     break;
336
337 case GET_ACC:
338     Acc( );
339
340
341
342

```

```

343         break;
344
345     case GET_MIN_SPEED:
346         Minspeed( );
347         break;
348
349     case GET_MAX_SPEED:
350         Maxspeed( );
351         break;
352
353
354     case GET_STP_REV:
355         SptRevol();
356         break;
357
358
359     case GET_R_MIN_SPD:
360         RMinSpd();
361         break;
362
363     case GET_R_MAX_SPD:
364         RMaxSpd();
365         break;
366
367     case GET_R_ACC:
368         RAcc();
369         break;
370
371
372
373     case STOPF:
374         Stop_Motion_fast();
375         Stop();
376         break;
377
378     case STOPN:
379         Stop_Motion_normal();
380         Stop();
381         break;
382
383     case CONSTSPD:
384         pch = strstr( message, "clock" );
385         if ( pch != NULL )
386         {
387             Way_Speed(STEP_CLOCKWISE);
388             putstring_P( PSTR( "DONE\r " ) );
389             break;
390         }
391
392
393         pch = strstr( message, "against" );
394         if ( pch != NULL )
395         {
396
397             Way_Speed( STEP_COUNTER_CLOCKWISE );
398             putstring_P( PSTR( "DONE\r" ) );
399             break;
400         }
401
402         Errorrmssg();
403         break;
404
405
406     case ENABLE:
407         Enable_Stepper( );
408         //putstring_P( PSTR( "Enable\r" ) );
409         break;
410
411     case DISABLE:
412         Disabled_Stepper( );
413         //putstring_P( PSTR( "Disable\r" ) );
414         break;
415
416
417
418
419
420     case GOTO:
421         pch = strstr( message, "clock" );
422         if ( pch != NULL )
423         {
424             pch = strstr( pch, " " );
425             ch = ConvertASCITouint64( pch );
426
427             if ( ch > 0 )
428             {
429                 Count_Step( STEP_CLOCKWISE, ch );

```



```

430 /*
431         putstring_P( PSTR( "DONE " ) );
432
433         putstring( int_to_string( ch ) );
434
435         putstring_P( PSTR( " steps\r" ) );
436
437 */
438         break;
439     } else
440     {
441         Errormssg( );
442         break;
443     }
444 }
445
446 pch = strstr( message, "against" );
447 if ( pch != NULL )
448 {
449     pch = strstr( pch, " " );
450     ch = ConvertASCITouint64( pch );
451     if ( ch > 0 )
452     {
453         Count_Step( STEP_COUNTER_CLOCKWISE, ch );
454     }
455 }
456 /*
457         putstring_P( PSTR( "DONE " ) );
458
459         putstring( int_to_string( ch ) );
460
461         putstring_P( PSTR( " steps\r" ) );
462
463 */
464         break;
465     } else
466     {
467         Errormssg( );
468         break;
469     }
470 }
471 break;
472
473 case STATUS:
474     switch ( Get_Decay( ) )
475     {
476     case SLOW_DECAY:
477         SDec( );
478         break;
479
480     case FAST_DECAY:
481         FDec( );
482         break;
483
484     case AUTO_DECAY:
485         ADec( );
486         break;
487
488     default:
489     {
490         //putstring_P( PSTR( "Not set decay\r" ) );
491         break;
492     }
493     }
494 }
495
496 switch ( Get_Mode( ) )
497 {
498     case MODE_FULL_STEP:
499         Fstep( );
500         break;
501
502     case MODE_HALF_STEP:
503         HStep( );
504         break;
505
506     case MODE_QUATER_STEP:
507         Qstep( );
508         break;
509
510     case MODE_8_MICROSTEP:
511         M8Step( );
512         break;
513
514     case MODE_16_MICROSTEP:
515         M16Step( );
516         break;
517
518     case MODE_32_MICROSTEP:
519         M32Step( );
520         break;
521 }

```

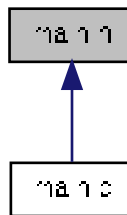
```
511             break;
512
513         default:
514         {
515             //putstring_P( PSTR( "Not set mode\r" ) );
516             break;
517         }
518
519     }
520
521
522     if ( Get_Start( ) )
523     {
524         //putstring_P( PSTR( "Enable\r" ) );
525     }
526     else
527     {
528         //putstring_P( PSTR( "Disable\r" ) );
529     }
530
531     break;
532
533
534
535
536
537
538
539     default:
540
541         if ( ( cmd < 0 ) && ( !( ( message[0] == 0x0d ) || ( message[0] == 0x0a ) ) ) )
542             putstring_P( PSTR( "Comand not implement\r" ) );
543         break;
544
545     } //switch (cmd)
546 } //if (USART_RxEOl > 0)
547
548
549
550
551 } //for (;;)
552
553 return 0;
554 } //End int main(void)
```


Set function for standart IO getchart and putchart from UART library

Definition at line 56 of file main.c.

7.7 main.h File Reference

This graph shows which files directly or indirectly include this file:



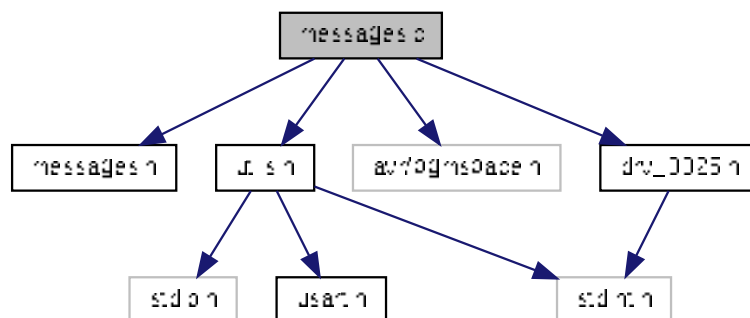
Functions

- void [Init_Input_Output](#) (void)
- void [test](#) (void)

7.8 messages.c File Reference

```
#include "messages.h"
#include "utils.h"
#include <avr/pgmspace.h>
#include "drv_8825.h"
```

Include dependency graph for messages.c:

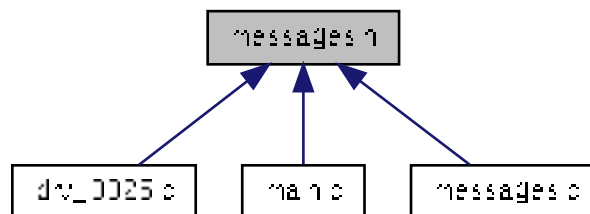


Functions

- void [Acc](#) (void)
- void [ADec](#) (void)
- void [Errormssg](#) (void)
- void [FDec](#) (void)
- void [Fstep](#) (void)
- void [HStep](#) (void)
- void [M16Step](#) (void)
- void [M32Step](#) (void)
- void [M8Step](#) (void)
- void [Maxspeed](#) (void)
- void [Minspeed](#) (void)
- void [Qstep](#) (void)
- void [RAcc](#) (void)
- void [RMaxSpd](#) (void)
- void [RMinSpd](#) (void)
- void [SDec](#) (void)
- void [SptRevol](#) (void)
- void [Stop](#) (void)

7.9 messages.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

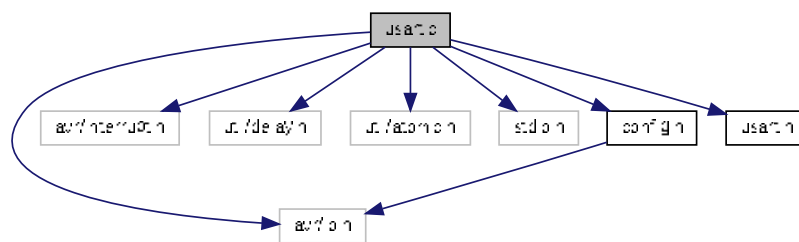
- void [Acc](#) (void)
- void [ADec](#) (void)
- void [Errormssg](#) (void)
- void [FDec](#) (void)
- void [Fstep](#) (void)
- void [HStep](#) (void)
- void [M16Step](#) (void)
- void [M32Step](#) (void)
- void [M8Step](#) (void)
- void [Maxspeed](#) (void)
- void [Minspeed](#) (void)

- void [Qstep](#) (void)
- void [RAcc](#) (void)
- void [RMaxSpd](#) (void)
- void [RMinSpd](#) (void)
- void [SDec](#) (void)
- void [SptRevol](#) (void)
- void [Stop](#) (void)

7.10 usart.c File Reference

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <util/atomic.h>
#include <stdio.h>
#include "config.h"
#include "usart.h"
```

Include dependency graph for usart.c:



Data Structures

- struct [ring_buffer](#)

Macros

- [#define B1SIZE](#) 32
- [#define BSIZE](#) 32

Typedefs

- typedef struct [ring_buffer](#) [ring_buffer_t](#)

Functions

- [ISR](#) (USART0_RXC_vect)
- [ISR](#) (USART0_UDRE_vect)
- int [usart_getchar](#) (FILE *stream)
Get received byte from ringbuffer.
- void [usart_init](#) (unsigned int baudrate)

Initialize USART and set baudrate.

- int `usart_putchar` (char c, FILE *stream)
Put byte to ringbuffer for transmitting via UART.

Variables

- static uint8_t `brx` [BSIZE]
- static uint8_t `btx` [BSIZE]
- volatile `ring_buffer_t` `usart_rx`
- volatile `ring_buffer_t` `usart_tx`

7.10.1 Macro Definition Documentation

7.10.1.1 #define B1SIZE 32

Definition at line 37 of file usart.c.

7.10.1.2 #define BSIZE 32

Definition at line 36 of file usart.c.

7.10.2 Typedef Documentation

7.10.2.1 typedef struct ring_buffer ring_buffer_t

7.10.3 Function Documentation

7.10.3.1 ISR (USART0_RXC_vect)

USART0 Receive Complete interrupt

Definition at line 195 of file usart.c.

```

196 {
197     uint8_t data;
198
199     data = UDR0;
200
201     if (usart_rx.fillcount < usart_rx.size)
202     {
203         if (data == VEOL)
204             usart_rx.nlines++;
205
206         usart_rx.data[usart_rx.head] = data;
207
208         usart_rx.head = (usart_rx.head + 1) & (usart_rx.
size - 1);
209         usart_rx.fillcount++;
210     }
211     else
212     {
213         //The Buffer is full! clear it
214         usart_rx.head = 0;
215         usart_rx.tail = 0;
216         usart_rx.fillcount = 0;
217         usart_rx.nlines = 0;
218
219         /* Wait for empty transmit buffer */
220         while (!(UCSR0A & (1 << UDRE0)))
221             ;
222         UDR0 = 'B';
223         while (!(UCSR0A & (1 << UDRE0)))
224             ;
225         UDR0 = 'O';
226         while (!(UCSR0A & (1 << UDRE0)))

```

```

227         ;
228         UDR0 = 'F';
229         while (!(UCSR0A & (1 << UDRE0)))
230             ;
231         UDR0 = VEOL;
232         //TODO Software flow control XON/XOFF.
233         http://www-user.tu-chemnitz.de/~heha/hs_freeware/terminal/terminal.htm
234         //SBUF = 0x13; //Pause transmission. Send XOFF character
235     }

```

7.10.3.2 ISR (USART0_UDRE_vect)

USART0 Data Register Empty interrupt Called when the USART is ready to transmit the next byte

Definition at line 242 of file usart.c.

```

243 {
244     if (usart_tx.fillcount > 0)
245     {
246         UDR0 = usart_tx.data[usart_tx.tail];
247         usart_tx.tail = (usart_tx.tail + 1) & (usart_tx.
248 size - 1);
249         usart_tx.fillcount--;
250     }
251     else
252     {
253         /* tx buffer empty, disable UDRE interrupt */
254         UCSR0B &= ~(1 << UDRIE0);
255     }
256 }

```

7.10.4 Variable Documentation

7.10.4.1 uint8_t brx[BSIZE] [static]

Definition at line 52 of file usart.c.

7.10.4.2 uint8_t btx[BSIZE] [static]

Definition at line 53 of file usart.c.

7.10.4.3 volatile ring_buffer_t usart_rx

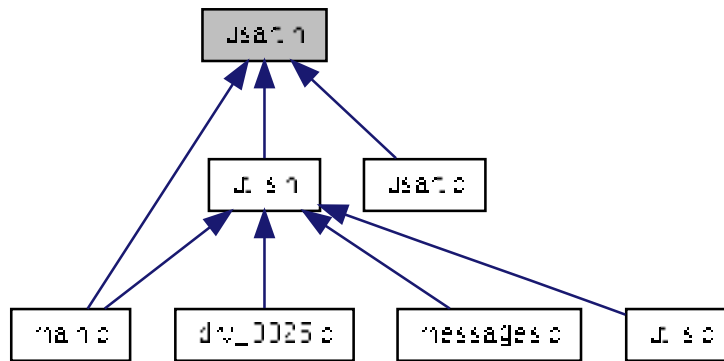
Definition at line 55 of file usart.c.

7.10.4.4 volatile ring_buffer_t usart_tx

Definition at line 54 of file usart.c.

7.11 usart.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define `USART_BAUD_SELECT`(baudRate, xtalCpu) (((xtalCpu)/((baudRate)*16))-1)
USART Baudrate Expression.
- #define `USART_BAUD_SELECT_DOUBLE_SPEED`(baudRate, xtalCpu) (((xtalCpu)/((baudRate)*8))-1)|0x8000)
USART Baudrate Expression for ATmega double speed mode.
- #define `VEOL` (char) 0x0d

Functions

- int `usart_getchar` (FILE *stream)
Get received byte from ringbuffer.
- void `usart_init` (unsigned int)
Initialize USART and set baudrate.
- int `usart_putchar` (char c, FILE *stream)
Put byte to ringbuffer for transmitting via UART.

7.12 usartm8.h File Reference

Macros

- #define `DOR0` DOR
- #define `FE0` FE
- #define `MPCM0` MPCM
- #define `RXB80` RXB8
- #define `RXC0` RXC
- #define `RXCIE0` RXCIE
- #define `RXEN0` RXEN

- #define [TXB80](#) TXB8
- #define [TXC0](#) TXC
- #define [TXCIE0](#) TXCIE
- #define [TXEN0](#) TXEN
- #define [U2X0](#) U2X
- #define [UBRR0H](#) UBRRH
- #define [UBRR0L](#) UBRRL
- #define [UBRR0L](#) UBRRL
- #define [UCPOL0](#) UCPOL
- #define [UCSR0A](#) UCSRA
- #define [UCSR0B](#) UCSRB
- #define [UCSR0C](#) UCSRC
- #define [UCSZ00](#) UCSZ0
- #define [UCSZ01](#) UCSZ1
- #define [UCSZ02](#) UCSZ2
- #define [UDR0](#) UDR
- #define [UDRE0](#) UDRE
- #define [UDRIE0](#) UDRIE
- #define [UMSEL0](#) UMSEL
- #define [UPE0](#) PE
- #define [UPM00](#) UPM0
- #define [UPM01](#) UPM1
- #define [URSEL0](#) URSEL
- #define [URSEL0](#) URSEL
- #define [USART0_RXC_vect](#) USART_RXC_vect
- #define [USART0_TXC_vect](#) USART_TXC_vect
- #define [USART0_UDRE_vect](#) USART_UDRE_vect
- #define [USARTXXX_H_](#) "usartm8.h"
- #define [USBS0](#) USBS

7.12.1 Macro Definition Documentation

7.12.1.1 #define DOR0 DOR

Definition at line 47 of file usartm8.h.

7.12.1.2 #define FE0 FE

Definition at line 46 of file usartm8.h.

7.12.1.3 #define MPCM0 MPCM

Definition at line 50 of file usartm8.h.

7.12.1.4 #define RXB80 RXB8

Definition at line 59 of file usartm8.h.

7.12.1.5 #define RXC0 RXC

Definition at line 43 of file usartm8.h.

7.12.1.6 #define RXCIE0 RXCIE

Definition at line 53 of file usartm8.h.

7.12.1.7 #define RXEN0 RXEN

Definition at line 56 of file usartm8.h.

7.12.1.8 #define TXB80 TXB8

Definition at line 60 of file usartm8.h.

7.12.1.9 #define TXC0 TXC

Definition at line 44 of file usartm8.h.

7.12.1.10 #define TXCIE0 TXCIE

Definition at line 54 of file usartm8.h.

7.12.1.11 #define TXEN0 TXEN

Definition at line 57 of file usartm8.h.

7.12.1.12 #define U2X0 U2X

Definition at line 49 of file usartm8.h.

7.12.1.13 #define UBRR0H UBRRH

Definition at line 32 of file usartm8.h.

7.12.1.14 #define UBRR0L UBRRL

Definition at line 33 of file usartm8.h.

7.12.1.15 #define UBRR1L UBRRL

Definition at line 33 of file usartm8.h.

7.12.1.16 #define UCPOL0 UCPOL

Definition at line 70 of file usartm8.h.

7.12.1.17 #define UCSRA UCSRA

Definition at line 30 of file usartm8.h.

7.12.1.18 #define UCSR0B UCSRB

Definition at line 29 of file usartm8.h.

7.12.1.19 #define UCSR0C UCSRC

Definition at line 28 of file usartm8.h.

7.12.1.20 #define UCSZ00 UCSZ0

Definition at line 69 of file usartm8.h.

7.12.1.21 #define UCSZ01 UCSZ1

Definition at line 68 of file usartm8.h.

7.12.1.22 #define UCSZ02 UCSZ2

Definition at line 58 of file usartm8.h.

7.12.1.23 #define UDR0 UDR

Definition at line 31 of file usartm8.h.

7.12.1.24 #define UDRE0 UDRE

Definition at line 45 of file usartm8.h.

7.12.1.25 #define UDRIE0 UDRIE

Definition at line 55 of file usartm8.h.

7.12.1.26 #define UMSEL0 UMSEL

Definition at line 64 of file usartm8.h.

7.12.1.27 #define UPE0 PE

Definition at line 48 of file usartm8.h.

7.12.1.28 #define UPM00 UPM0

Definition at line 66 of file usartm8.h.

7.12.1.29 #define UPM01 UPM1

Definition at line 65 of file usartm8.h.

7.12.1.30 `#define URSEL0 URSEL`

Definition at line 63 of file usartm8.h.

7.12.1.31 `#define URSEL0 URSEL`

Definition at line 63 of file usartm8.h.

7.12.1.32 `#define USART0_RXC_vect USART_RXC_vect`

Definition at line 35 of file usartm8.h.

7.12.1.33 `#define USART0_TXC_vect USART_TXC_vect`

Definition at line 36 of file usartm8.h.

7.12.1.34 `#define USART0_UDRE_vect USART_UDRE_vect`

Definition at line 37 of file usartm8.h.

7.12.1.35 `#define USARTXXX_H_ "usartm8.h"`

Definition at line 18 of file usartm8.h.

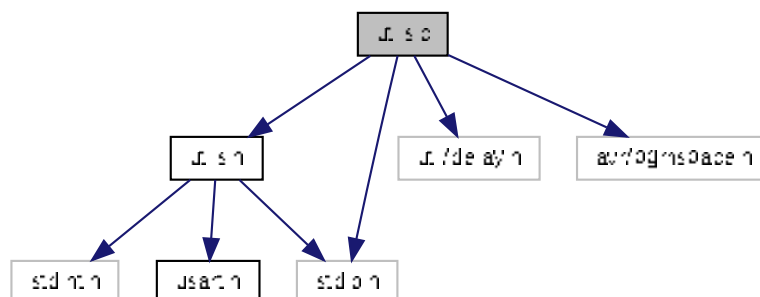
7.12.1.36 `#define USBS0 USBS`

Definition at line 67 of file usartm8.h.

7.13 utils.c File Reference

```
#include "utils.h"
#include <stdio.h>
#include <util/delay.h>
#include <avr/pgmspace.h>
```

Include dependency graph for utils.c:



Functions

- uint64_t [ConvertASCItouint64](#) (char *in)
- char * [int_to_string](#) (uint64_t i)
- void [putstring](#) (const char *putc)
- void [putstring_P](#) (const char *putc)

Variables

- static char [buf](#) [20]

7.13.1 Variable Documentation

7.13.1.1 char buf[20] [static]

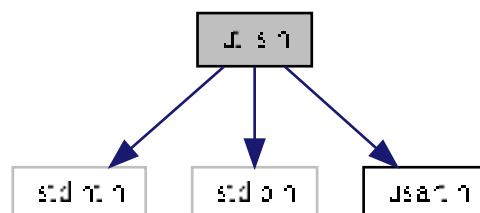
Temporary buffer for converting uint64_t to string

Definition at line 37 of file utils.c.

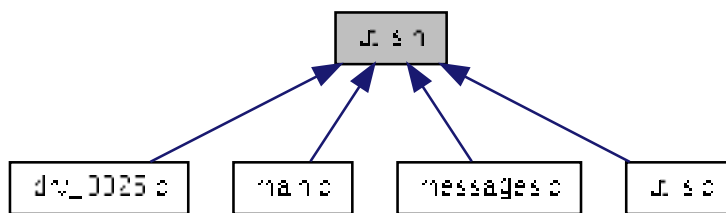
7.14 utils.h File Reference

```
#include <stdint.h>
#include <stdio.h>
#include "usart.h"
```

Include dependency graph for utils.h:



This graph shows which files directly or indirectly include this file:



Functions

- uint64_t [ConvertASCltouint64](#) (char *in)
- char * [int_to_string](#) (uint64_t i)
- void [putstring](#) (const char *putc)
- void [putstring_P](#) (const char *putc)

Index

AUTO_DECAY
 DRV8825 Library, [18](#)

ADec
 messages Library, [51](#)

Acc
 messages Library, [50](#)

Acceleration
 Motor_Parameters, [80](#)

B1SIZE
 usart.c, [109](#)

BSIZE
 usart.c, [109](#)

brx
 usart.c, [110](#)

btx
 usart.c, [110](#)

buf
 utils.c, [116](#)

CONSTSPD
 interpreter.h, [94](#)

CHECKPIN
 config Library, [11](#)

CalWorkParam
 DRV8825 Library, [19](#)

check_pin
 config Library, [11](#)

clr_bit
 config Library, [11](#)

clr_port
 config Library, [11](#)

cmd_struct, [77](#)
 id, [77](#)
 name, [78](#)

cmd_tbl
 interpreter.c, [93](#)

config Library, [9](#)
 CHECKPIN, [11](#)
 check_pin, [11](#)
 clr_bit, [11](#)
 clr_port, [11](#)
 DDECAY, [11](#)
 DDIR_485, [11](#)
 DDIRECTION, [11](#)
 DECAY, [11](#)
 DENABLE_STEPPER, [11](#)
 DFAULT, [11](#)
 DHOME, [11](#)
 DIR_485, [11](#)
 DIRECTION, [12](#)
 DMISO, [12](#)
 DMODE0, [12](#)
 DMODE1, [12](#)
 DMODE2, [12](#)
 DMOSI, [12](#)
 DRXD, [12](#)
 DSCL, [12](#)
 DSTEP, [12](#)
 DTXD, [12](#)
 ENABLE_STEPPER, [12](#)
 FAULT, [13](#)
 false, [12](#)
 HOME, [13](#)
 IDECAY, [13](#)
 IDIR_485, [13](#)
 IDIRECTION, [13](#)
 IENABLE_STEPPER, [13](#)
 IFault, [13](#)
 IHOME, [13](#)
 IMISO, [13](#)
 IMODE0, [13](#)
 IMODE1, [13](#)
 IMODE2, [13](#)
 IMOSI, [14](#)
 IRXD, [14](#)
 ISCL, [14](#)
 ISTEP, [14](#)
 ITXD, [14](#)
 MISO, [14](#)
 MODE0, [14](#)
 MODE1, [14](#)
 MODE2, [14](#)
 MOSI, [14](#)
 PDECAY, [14](#)
 PDIR_485, [14](#)
 PDIRECTION, [15](#)
 PENABLE_STEPPER, [15](#)
 PFAULT, [15](#)
 PHOME, [15](#)
 PMISO, [15](#)
 PMODE0, [15](#)
 PMODE1, [15](#)
 PMODE2, [15](#)
 PMOSI, [15](#)
 PRXD, [15](#)
 PSCL, [15](#)
 PSTEP, [15](#)
 PTXD, [16](#)

- RXD, [16](#)
- SCL, [16](#)
- STEP, [16](#)
- set_as_input, [16](#)
- set_as_output, [16](#)
- set_bit, [16](#)
- set_port, [16](#)
- TXD, [16](#)
- true, [16](#)
- config.h, [83](#)
- ConstSpd
 - drv_8825.c, [88](#)
- ConvertASCIItouint64
 - utils Library, [70](#)
- Count_Step
 - DRV8825 Library, [20](#)
- CurrentDecay
 - drv_8825.c, [88](#)
- CurrentMode
 - drv_8825.c, [88](#)
- CurrentSpeed
 - Motion, [79](#)
- DISABLE
 - interpreter.h, [94](#)
- DRV8825 Library
 - AUTO_DECAY, [18](#)
 - FAST_DECAY, [18](#)
 - MODE_16_MICROSTEP, [18](#)
 - MODE_32_MICROSTEP, [18](#)
 - MODE_8_MICROSTEP, [18](#)
 - MODE_FULL_STEP, [18](#)
 - MODE_HALF_STEP, [18](#)
 - MODE_QUATER_STEP, [18](#)
 - SLOW_DECAY, [18](#)
 - STEP_CLOCKWISE, [19](#)
 - STEP_COUNTER_CLOCKWISE, [19](#)
- DDECAY
 - config Library, [11](#)
- DDIR_485
 - config Library, [11](#)
- DDIRECTION
 - config Library, [11](#)
- DECAY
 - config Library, [11](#)
- DENABLE_STEPPER
 - config Library, [11](#)
- DFAULT
 - config Library, [11](#)
- DHOME
 - config Library, [11](#)
- DIR_485
 - config Library, [11](#)
- DIRECTION
 - config Library, [12](#)
- DMISO
 - config Library, [12](#)
- DMODE0
 - config Library, [12](#)
- DMODE1
 - config Library, [12](#)
- DMODE2
 - config Library, [12](#)
- DMOSI
 - config Library, [12](#)
- DOR0
 - usartm8.h, [112](#)
- DRV8825 Library, [17](#)
 - CalWorkParam, [19](#)
 - Count_Step, [20](#)
 - Decay, [21](#)
 - decay_type, [18](#)
 - Disabled_Stepper, [22](#)
 - Enable_Stepper, [23](#)
 - Get_Decay, [23](#)
 - Get_Mode, [24](#)
 - Get_Start, [24](#)
 - Get_Steps_revol, [25](#)
 - GetAcceleration, [25](#)
 - GetCurrentSpeed, [26](#)
 - GetMaxSpeed, [26](#)
 - GetMinSpeed, [27](#)
 - GetRealAcc, [27](#)
 - GetRealMaxSpeed, [28](#)
 - GetRealMinSpeed, [28](#)
 - HelperRtoT, [29](#)
 - HelperTtoR, [30](#)
 - Home, [31](#)
 - InitStepper, [31](#)
 - Mode, [33](#)
 - mode_type, [18](#)
 - RealToTime, [34](#)
 - Set_Steps_revol, [35](#)
 - SetAcceleration, [36](#)
 - SetCurrentSpeed, [37](#)
 - SetMaxSpeed, [37](#)
 - SetMinSpeed, [38](#)
 - SetRealAcc, [39](#)
 - SetRealMaxSpeed, [39](#)
 - SetRealMinSpeed, [40](#)
 - Step, [41](#)
 - step_type, [18](#)
 - Stop_Motion_fast, [42](#)
 - Stop_Motion_normal, [42](#)
 - store, [43](#)
 - TimeToReal, [44](#)
 - Way_Speed, [46](#)
- DRXD
 - config Library, [12](#)
- DSCL
 - config Library, [12](#)
- DSTEP
 - config Library, [12](#)
- DTXD
 - config Library, [12](#)
- data
 - ring_buffer, [82](#)

- Decay
 - DRV8825 Library, [21](#)
- decay_type
 - DRV8825 Library, [18](#)
- Disabled_Stepper
 - DRV8825 Library, [22](#)
- drv_8825.c, [85](#)
 - ConstSpd, [88](#)
 - CurrentDecay, [88](#)
 - CurrentMode, [88](#)
 - ISR, [87](#)
 - ModeStart, [88](#)
 - Motion, [88](#)
 - Motor_Parameters, [87](#)
 - PERTMR, [87](#)
 - RealSpeed, [88](#)
 - StepDir, [88](#)
 - StepsPerRev, [89](#)
 - TIMERCONST, [87](#)
 - TimeParam, [89](#)
 - WorkingParam, [89](#)
- drv_8825.h, [89](#)
- ENABLE
 - interpreter.h, [94](#)
- ENABLE_STEPPER
 - config Library, [12](#)
- Enable_Stepper
 - DRV8825 Library, [23](#)
- Errormssg
 - messages Library, [52](#)
- FAST_DECAY
 - DRV8825 Library, [18](#)
- F_CPU
 - main.c, [98](#)
- FAULT
 - config Library, [13](#)
- FDec
 - messages Library, [52](#)
- FE0
 - usartm8.h, [112](#)
- false
 - config Library, [12](#)
- fillcount
 - ring_buffer, [82](#)
- Fstep
 - messages Library, [53](#)
- GET_ACC
 - interpreter.h, [94](#)
- GET_MAX_SPEED
 - interpreter.h, [94](#)
- GET_MIN_SPEED
 - interpreter.h, [94](#)
- GET_R_ACC
 - interpreter.h, [94](#)
- GET_R_MAX_SPD
 - interpreter.h, [94](#)
- GET_R_MIN_SPD
 - interpreter.h, [94](#)
- GET_STP_REV
 - interpreter.h, [94](#)
- GOTO
 - interpreter.h, [94](#)
- Get_Decay
 - DRV8825 Library, [23](#)
- Get_Mode
 - DRV8825 Library, [24](#)
- Get_Start
 - DRV8825 Library, [24](#)
- Get_Steps_revol
 - DRV8825 Library, [25](#)
- get_cmd_id
 - interpreter.c, [91](#)
 - interpreter.h, [95](#)
- get_cmd_name
 - interpreter.c, [92](#)
 - interpreter.h, [95](#)
- GetAcceleration
 - DRV8825 Library, [25](#)
- GetCurrentSpeed
 - DRV8825 Library, [26](#)
- GetMaxSpeed
 - DRV8825 Library, [26](#)
- GetMinSpeed
 - DRV8825 Library, [27](#)
- GetRealAcc
 - DRV8825 Library, [27](#)
- GetRealMaxSpeed
 - DRV8825 Library, [28](#)
- GetRealMinSpeed
 - DRV8825 Library, [28](#)
- HELP
 - interpreter.h, [94](#)
- HOME
 - config Library, [13](#)
- HStep
 - messages Library, [54](#)
- head
 - ring_buffer, [82](#)
- HelperRtoT
 - DRV8825 Library, [29](#)
- HelperTtoR
 - DRV8825 Library, [30](#)
- Home
 - DRV8825 Library, [31](#)
- IDECAY
 - config Library, [13](#)
- IDIR_485
 - config Library, [13](#)
- IDIRECTION
 - config Library, [13](#)
- IENABLE_STEPPER
 - config Library, [13](#)
- IFault

- config Library, 13
- IHOME
 - config Library, 13
- IMISO
 - config Library, 13
- IMODE0
 - config Library, 13
- IMODE1
 - config Library, 13
- IMODE2
 - config Library, 13
- IMOSI
 - config Library, 14
- IRXD
 - config Library, 14
- ISCL
 - config Library, 14
- ISR
 - drv_8825.c, 87
 - usart.c, 109, 110
- ISTEP
 - config Library, 14
- ITXD
 - config Library, 14
- id
 - cmd_struct, 77
- Init_Input_Output
 - main Library, 48
- InitStepper
 - DRV8825 Library, 31
- int_to_string
 - utils Library, 71
- interpreter.h
 - CONSTSPD, 94
 - DISABLE, 94
 - ENABLE, 94
 - GET_ACC, 94
 - GET_MAX_SPEED, 94
 - GET_MIN_SPEED, 94
 - GET_R_ACC, 94
 - GET_R_MAX_SPD, 94
 - GET_R_MIN_SPD, 94
 - GET_STP_REV, 94
 - GOTO, 94
 - HELP, 94
 - SET_ACC, 94
 - SET_DEC_AUTO, 94
 - SET_DEC_FAST, 94
 - SET_DEC_SLOW, 94
 - SET_HOME, 94
 - SET_MAX_SPEED, 94
 - SET_MIN_SPEED, 94
 - SET_MODE_16, 94
 - SET_MODE_32, 94
 - SET_MODE_8, 94
 - SET_MODE_FULL, 94
 - SET_MODE_HALF, 94
 - SET_MODE_QUATER, 94
 - SET_R_ACC, 94
 - SET_R_MAX_SPD, 94
 - SET_R_MIN_SPD, 94
 - SET_STP_REV, 94
 - STATUS, 94
 - STOPF, 94
 - STOPN, 94
 - TEST, 94
 - VERSION, 94
- interpreter.c, 91
 - cmd_tbl, 93
 - get_cmd_id, 91
 - get_cmd_name, 92
- interpreter.h, 93
 - get_cmd_id, 95
 - get_cmd_name, 95
- M16Step
 - messages Library, 54
- M32Step
 - messages Library, 55
- M8Step
 - messages Library, 55
- MODE_16_MICROSTEP
 - DRV8825 Library, 18
- MODE_32_MICROSTEP
 - DRV8825 Library, 18
- MODE_8_MICROSTEP
 - DRV8825 Library, 18
- MODE_FULL_STEP
 - DRV8825 Library, 18
- MODE_HALF_STEP
 - DRV8825 Library, 18
- MODE_QUATER_STEP
 - DRV8825 Library, 18
- MISO
 - config Library, 14
- MODE0
 - config Library, 14
- MODE1
 - config Library, 14
- MODE2
 - config Library, 14
- MOSI
 - config Library, 14
- MPCM0
 - usartm8.h, 112
- main
 - main.c, 98
- main Library, 48
 - Init_Input_Output, 48
 - test, 49
- main.c, 97
 - F_CPU, 98
 - main, 98
 - USART_BAUD_RATE, 98
 - usart_str, 105
- main.h, 106
- MaxSpeed

- Motor_Parameters, [80](#)
- Maxspeed
 - messages Library, [56](#)
- messages Library, [50](#)
 - ADec, [51](#)
 - Acc, [50](#)
 - Errormsg, [52](#)
 - FDec, [52](#)
 - Fstep, [53](#)
 - HStep, [54](#)
 - M16Step, [54](#)
 - M32Step, [55](#)
 - M8Step, [55](#)
 - Maxspeed, [56](#)
 - Minspeed, [57](#)
 - Qstep, [58](#)
 - RAcc, [59](#)
 - RMaxSpd, [59](#)
 - RMinSpd, [60](#)
 - SDec, [61](#)
 - SptRevol, [62](#)
 - Stop, [62](#)
- messages.c, [106](#)
- messages.h, [107](#)
- MinSpeed
 - Motor_Parameters, [80](#)
- Minspeed
 - messages Library, [57](#)
- Mode
 - DRV8825 Library, [33](#)
- mode_type
 - DRV8825 Library, [18](#)
- ModeStart
 - drv_8825.c, [88](#)
- Motion, [78](#)
 - CurrentSpeed, [79](#)
 - drv_8825.c, [88](#)
 - StepsAccel, [79](#)
 - StepsConstSpeed, [79](#)
 - StepsDeaccel, [79](#)
- Motor_Parameters, [79](#)
 - Acceleration, [80](#)
 - drv_8825.c, [87](#)
 - MaxSpeed, [80](#)
 - MinSpeed, [80](#)
- name
 - cmd_struct, [78](#)
- nlines
 - ring_buffer, [82](#)
- PDECAY
 - config Library, [14](#)
- PDIR_485
 - config Library, [14](#)
- PDIRECTION
 - config Library, [15](#)
- PENABLE_STEPPER
 - config Library, [15](#)
- PERTMR
 - drv_8825.c, [87](#)
- PFAULT
 - config Library, [15](#)
- PHOME
 - config Library, [15](#)
- PMISO
 - config Library, [15](#)
- PMODE0
 - config Library, [15](#)
- PMODE1
 - config Library, [15](#)
- PMODE2
 - config Library, [15](#)
- PMOSI
 - config Library, [15](#)
- PRXD
 - config Library, [15](#)
- PSCL
 - config Library, [15](#)
- PSTEP
 - config Library, [15](#)
- PTXD
 - config Library, [16](#)
- putstring
 - utils Library, [72](#)
- putstring_P
 - utils Library, [73](#)
- Qstep
 - messages Library, [58](#)
- RAcc
 - messages Library, [59](#)
- RMaxSpd
 - messages Library, [59](#)
- RMinSpd
 - messages Library, [60](#)
- RXB80
 - usartm8.h, [112](#)
- RXC0
 - usartm8.h, [112](#)
- RXCIE0
 - usartm8.h, [112](#)
- RXD
 - config Library, [16](#)
- RXEN0
 - usartm8.h, [113](#)
- RealSpeed
 - drv_8825.c, [88](#)
- RealToTime
 - DRV8825 Library, [34](#)
- ring_buffer, [81](#)
 - data, [82](#)
 - fillcount, [82](#)
 - head, [82](#)
 - nlines, [82](#)
 - size, [82](#)
 - tail, [82](#)

- ring_buffer_t
 - usart.c, [109](#)
- SET_ACC
 - interpreter.h, [94](#)
- SET_DEC_AUTO
 - interpreter.h, [94](#)
- SET_DEC_FAST
 - interpreter.h, [94](#)
- SET_DEC_SLOW
 - interpreter.h, [94](#)
- SET_HOME
 - interpreter.h, [94](#)
- SET_MAX_SPEED
 - interpreter.h, [94](#)
- SET_MIN_SPEED
 - interpreter.h, [94](#)
- SET_MODE_16
 - interpreter.h, [94](#)
- SET_MODE_32
 - interpreter.h, [94](#)
- SET_MODE_8
 - interpreter.h, [94](#)
- SET_MODE_FULL
 - interpreter.h, [94](#)
- SET_MODE_HALF
 - interpreter.h, [94](#)
- SET_MODE_QUATER
 - interpreter.h, [94](#)
- SET_R_ACC
 - interpreter.h, [94](#)
- SET_R_MAX_SPD
 - interpreter.h, [94](#)
- SET_R_MIN_SPD
 - interpreter.h, [94](#)
- SET_STP_REV
 - interpreter.h, [94](#)
- SLOW_DECAY
 - DRV8825 Library, [18](#)
- STATUS
 - interpreter.h, [94](#)
- STEP_CLOCKWISE
 - DRV8825 Library, [19](#)
- STEP_COUNTER_CLOCKWISE
 - DRV8825 Library, [19](#)
- STOPF
 - interpreter.h, [94](#)
- STOPN
 - interpreter.h, [94](#)
- SCL
 - config Library, [16](#)
- SDec
 - messages Library, [61](#)
- STEP
 - config Library, [16](#)
- Set_Steps_revol
 - DRV8825 Library, [35](#)
- set_as_input
 - config Library, [16](#)
- set_as_output
 - config Library, [16](#)
- set_bit
 - config Library, [16](#)
- set_port
 - config Library, [16](#)
- SetAcceleration
 - DRV8825 Library, [36](#)
- SetCurrentSpeed
 - DRV8825 Library, [37](#)
- SetMaxSpeed
 - DRV8825 Library, [37](#)
- SetMinSpeed
 - DRV8825 Library, [38](#)
- SetRealAcc
 - DRV8825 Library, [39](#)
- SetRealMaxSpeed
 - DRV8825 Library, [39](#)
- SetRealMinSpeed
 - DRV8825 Library, [40](#)
- size
 - ring_buffer, [82](#)
- SptRevol
 - messages Library, [62](#)
- Step
 - DRV8825 Library, [41](#)
- step_type
 - DRV8825 Library, [18](#)
- StepDir
 - drv_8825.c, [88](#)
- StepsAccel
 - Motion, [79](#)
- StepsConstSpeed
 - Motion, [79](#)
- StepsDeaccel
 - Motion, [79](#)
- StepsPerRev
 - drv_8825.c, [89](#)
- Stop
 - messages Library, [62](#)
- Stop_Motion_fast
 - DRV8825 Library, [42](#)
- Stop_Motion_normal
 - DRV8825 Library, [42](#)
- store
 - DRV8825 Library, [43](#)
- TEST
 - interpreter.h, [94](#)
- TIMERCONST
 - drv_8825.c, [87](#)
- TXB80
 - usartm8.h, [113](#)
- TXC0
 - usartm8.h, [113](#)
- TXCIE0
 - usartm8.h, [113](#)
- TXD
 - config Library, [16](#)

TXEN0
 usartm8.h, 113
 tail
 ring_buffer, 82
 test
 main Library, 49
 TimeParam
 drv_8825.c, 89
 TimeToReal
 DRV8825 Library, 44
 true
 config Library, 16
 U2X0
 usartm8.h, 113
 UART Library, 64
 USART_BAUD_SELECT, 64
 usart_getchar, 65
 usart_init, 66
 usart_putchar, 67
 VEOL, 65
 UBRR0H
 usartm8.h, 113
 UBRR0L
 usartm8.h, 113
 UCPOL0
 usartm8.h, 113
 UCSR0A
 usartm8.h, 113
 UCSR0B
 usartm8.h, 113
 UCSR0C
 usartm8.h, 114
 UCSZ00
 usartm8.h, 114
 UCSZ01
 usartm8.h, 114
 UCSZ02
 usartm8.h, 114
 UDR0
 usartm8.h, 114
 UDRE0
 usartm8.h, 114
 UDRIE0
 usartm8.h, 114
 UMSEL0
 usartm8.h, 114
 UPE0
 usartm8.h, 114
 UPM00
 usartm8.h, 114
 UPM01
 usartm8.h, 114
 URSEL0
 usartm8.h, 114, 115
 USART0_RXC_vect
 usartm8.h, 115
 USART0_TXC_vect
 usartm8.h, 115
 USART0_UDRE_vect
 usartm8.h, 115
 USART_BAUD_RATE
 main.c, 98
 USART_BAUD_SELECT
 UART Library, 64
 USARTXXX_H_
 usartm8.h, 115
 USB0
 usartm8.h, 115
 usart.c, 108
 B1SIZE, 109
 BSIZE, 109
 brx, 110
 btx, 110
 ISR, 109, 110
 ring_buffer_t, 109
 usart_rx, 110
 usart_tx, 110
 usart.h, 111
 usart_getchar
 UART Library, 65
 usart_init
 UART Library, 66
 usart_putchar
 UART Library, 67
 usart_rx
 usart.c, 110
 usart_str
 main.c, 105
 usart_tx
 usart.c, 110
 usartm8.h, 111
 DOR0, 112
 FE0, 112
 MPCM0, 112
 RXB80, 112
 RXC0, 112
 RXCIE0, 112
 RXEN0, 113
 TXB80, 113
 TXC0, 113
 TXCIE0, 113
 TXEN0, 113
 U2X0, 113
 UBRR0H, 113
 UBRR0L, 113
 UCPOL0, 113
 UCSR0A, 113
 UCSR0B, 113
 UCSR0C, 114
 UCSZ00, 114
 UCSZ01, 114
 UCSZ02, 114
 UDR0, 114
 UDRE0, 114
 UDRIE0, 114
 UMSEL0, 114

- UPE0, [114](#)
- UPM00, [114](#)
- UPM01, [114](#)
- URSEL0, [114](#), [115](#)
- USART0_RXC_vect, [115](#)
- USART0_TXC_vect, [115](#)
- USART0_UDRE_vect, [115](#)
- USARTXXX_H_, [115](#)
- USBS0, [115](#)
- utils Library, [70](#)
 - ConvertASCItouint64, [70](#)
 - int_to_string, [71](#)
 - putstring, [72](#)
 - putstring_P, [73](#)
- utils.c, [115](#)
 - buf, [116](#)
- utils.h, [116](#)
- VERSION
 - interpreter.h, [94](#)
- VEOL
 - UART Library, [65](#)
- Way_Speed
 - DRV8825 Library, [46](#)
- WorkingParam
 - drv_8825.c, [89](#)