Stepper motor controller

1.0

Generated by Doxygen 1.8.5

Sun Apr 19 2015 20:30:08

Contents

Chapter 1

Todo List

Global Home (void)

it doesnt work as espected

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Module Documentation

5.1 config Library

General macros for port manipolation 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 manipolation and port naming.

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

Defines simple macros for setting bits and bytes for manipolating 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.

```
        {SLOW_DECAY,       /**< Status for setting slow decay. */
    FAST_DECAY,            /**< Status for setting fast decay. */
    AUTO_DECAY             /**< Status for setting auto decay. */
    } 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.

```
        {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. */
    } 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

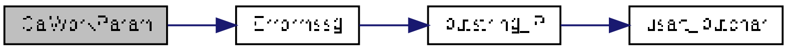
void

Definition at line 723 of file drv_8825.c.

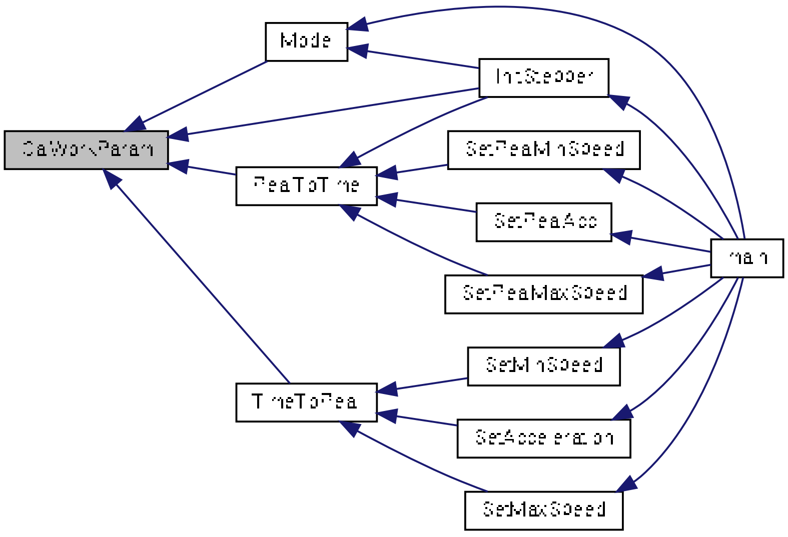
```
724 {
725     switch ( CurrentMode )
726     {
727         case MODE_FULL_STEP:
728             WorkingParam.MaxSpeed = TimeParam.
      MaxSpeed;
729             WorkingParam.MinSpeed = TimeParam.
      MinSpeed;
730             WorkingParam.Acceleration = TimeParam.
      Acceleration;
731             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;
            WorkingParam.MinSpeed = TimeParam.
  MinSpeed >> 4;
            WorkingParam.Acceleration = TimeParam.
  Acceleration >> 4;
            break;


        case MODE_32_MICROSTEP:
            WorkingParam.MaxSpeed = TimeParam.
  MaxSpeed >> 5;
            WorkingParam.MinSpeed = TimeParam.
  MinSpeed >> 5;
            WorkingParam.Acceleration = TimeParam.
  Acceleration >> 5;
            break;

        default:
            Errormssg( );

            break;
    }
}
```

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

step	the first argument.
step_count	the second argument.

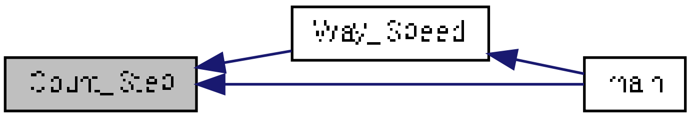
Returns

void

Definition at line 297 of file drv_8825.c.

```
{

    uint16_t y;
    StepDir = step;


    y = ( WorkingParam.MinSpeed - WorkingParam.MaxSpeed ) / WorkingParam.Acceleration;
    y = y << 1;
    if ( y < step_count )
    {
        Motion.StepsAccel = y >> 1;
        Motion.StepsDeaccel = Motion.StepsAccel;
        Motion.StepsConstSpeed = step_count - ( Motion.StepsAccel + Motion.StepsDeaccel );

    } else
    {
        Motion.StepsConstSpeed = 0;
        Motion.StepsAccel = step_count >> 1;
        Motion.StepsDeaccel = Motion.StepsAccel;
        if ( ( step_count & 1 ) )
        {
            Motion.StepsAccel++;
        }

    }
    clr_bit( TCCR1B, CS10 );
    set_bit( TCCR1B, CS11 );
    clr_bit( TCCR1B, CS12 );
    TCNT1 = 65535 - WorkingParam.MinSpeed;
    Motion.CurrentSpeed = WorkingParam.MinSpeed;
    set_bit( TIMSK, TOIE1 );

}
```

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

decay	the first argument.

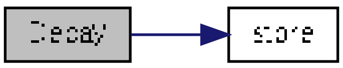
Returns

void

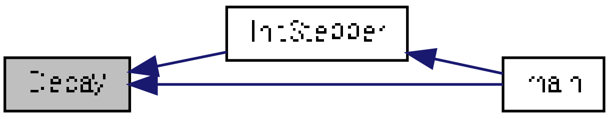
Definition at line 185 of file drv_8825.c.

```
{
    CurrentDecay = decay;
    store( );
    if ( decay == FAST_DECAY )
    {
        set_as_output( DDECAY, DECAY );
        set_bit( PDECAY, DECAY );
        return;
    }

    if ( decay == SLOW_DECAY )
    {
        set_as_output( DDECAY, DECAY );
        clr_bit( PDECAY, DECAY );
        return;
    }

    set_as_input( DDECAY, DECAY );
    clr_bit( PDECAY, DECAY );
}
```

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.

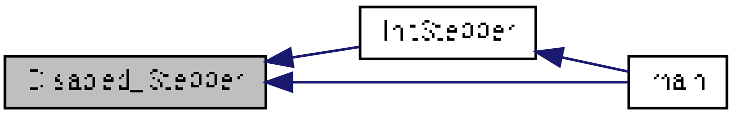
Returns

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.

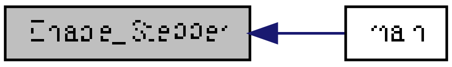
Returns

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

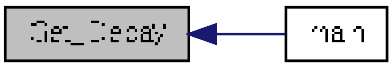
Returns

decay_type

Definition at line 395 of file drv_8825.c.

```
{
    return CurrentDecay;
}
```

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

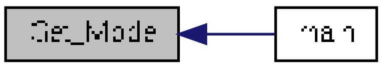
Returns

mode_type

Definition at line 405 of file drv_8825.c.

```
{
    return CurrentMode;
}
```

Here is the caller graph for this function:



5.2.3.8 uint8_t Get_Start (void)

Returns the mode.

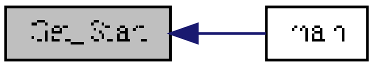
Returns

uint8_t

Definition at line 414 of file drv_8825.c.

```
{
    return ModeStart;
}
```

Here is the caller graph for this function:



5.2.3.9 uint16_t Get_Steps_revol (void)

Gets the steps per revolution.

See Also

StepsPerRev

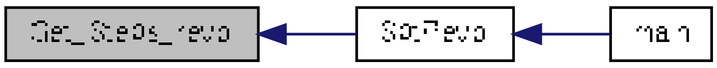
Returns

uint16_t

Definition at line 545 of file drv_8825.c.

```
{
    return StepsPerRev;
}
```

Here is the caller graph for this function:



5.2.3.10 uint16_t GetAcceleration (void)

Reads the setted acceleration.

See Also

TimeParam.Acceleration

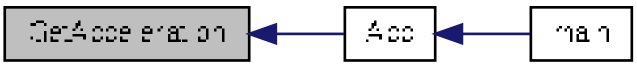
Returns

uint16_t

Definition at line 445 of file drv_8825.c.

```
{
    return TimeParam.Acceleration;
}
```

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.

```
{
    return Motion.CurrentSpeed;
}
```

5.2.3.12 uint16_t GetMaxSpeed (void)

Reads the setted maximum speed..

See Also

TimeParam.MaxSpeed

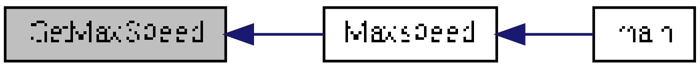
Returns

uint16_t

Definition at line 424 of file drv_8825.c.

```
{
    return TimeParam.MaxSpeed;
}
```

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

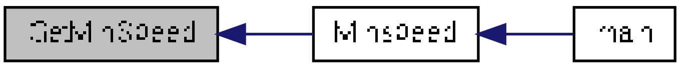
Returns

uint16_t

Definition at line 435 of file drv_8825.c.

```
{
    return TimeParam.MinSpeed;
}
```

Here is the caller graph for this function:



5.2.3.14 uint16_t GetRealAcc (void)

Gets the real acceleration.

See Also

RealSpeed.Acceleration

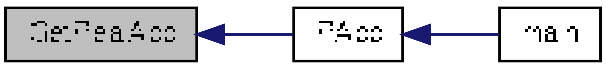
Returns

uint16_t

Definition at line 613 of file drv_8825.c.

```
{
    return RealSpeed.Acceleration;
}
```

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

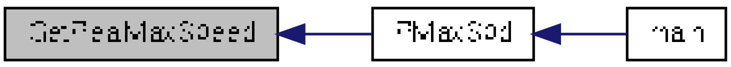
Returns

uint16_t

Definition at line 592 of file drv_8825.c.

```
{
    return RealSpeed.MaxSpeed;
}
```

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

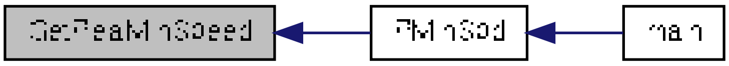
Returns

uint16_t

Definition at line 603 of file drv_8825.c.

```
{
    return RealSpeed.MinSpeed;
}
```

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

in	the first argument.

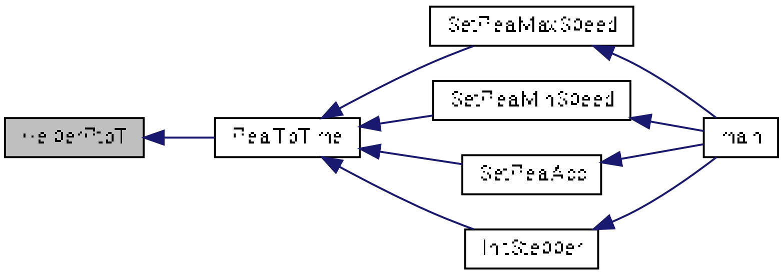
Returns

uint64_t

Definition at line 838 of file drv_8825.c.

```
{
    uint64_t revpermin;

/*
    revpermin = in;
    revpermin = revpermin*StepsPerRev;
    revpermin = revpermin / 60;
    revpermin = 1000000000 / revpermin;
    revpermin = revpermin / PERTMR;
*/
revpermin=TIMERCONST/(in*StepsPerRev);
    if ( revpermin < 65500 )
    {
        return revpermin;
    }
    else
    {
        return  65500;
    }
}
```

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

in	the first argument.

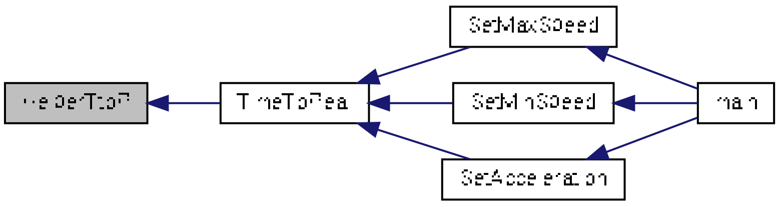
Returns

uint64_t

Definition at line 867 of file drv_8825.c.

```
{
    uint64_t r;

    //const uint64_t l=1000000000*60;

    r=TIMERCONST/(in*StepsPerRev);

/*
    r = in;



    r = r * PERTMR;
    r = r*StepsPerRev;
    r = 10000000000 / r;
    r = r * 6;

*/



    if ( r > 1 )
    {
        return r;
    } else
    {
        return 1;
        //putstring_P( PSTR( "Target MaxSpeed\r" ) );
    }

}
```

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 espected

See Also

step_type

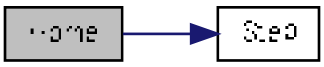
Returns

uint8_t

Definition at line 376 of file drv_8825.c.

```
{
    int16_t i;

    for ( i = 0; i < 128; i++ )
    {
        Step(STEP_COUNTER_CLOCKWISE);
        if ( !(check_pin( IHOME, HOME )) )
            return true;
    }
    return false;
}
```

Here is the call graph for this function:



5.2.3.20 void InitStepper (void)

Initialize the DRV232.

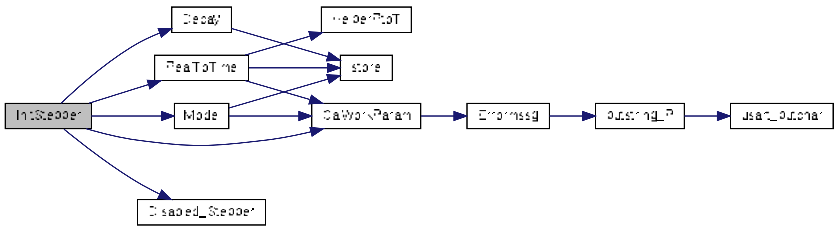
Returns

void

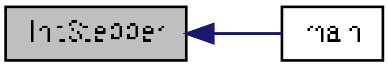
Definition at line 778 of file drv_8825.c.

```
{
    RealSpeed.MaxSpeed = eeprom_read_word( ( uint16_t * ) ( 0 ) );
    RealSpeed.MinSpeed = eeprom_read_word( ( uint16_t * ) ( 2 ) );
    RealSpeed.Acceleration = eeprom_read_word( ( uint16_t * ) ( 4 ) );
    StepsPerRev = eeprom_read_word( ( uint16_t * ) ( 6 ) );
    CurrentMode = eeprom_read_byte( ( uint8_t * ) ( 8 ) );
    CurrentDecay = eeprom_read_byte( ( uint8_t * ) ( 9 ) );
    if ( ( RealSpeed.MaxSpeed == 0 ) || ( RealSpeed.MaxSpeed == 0xffff ) )
    {
        RealSpeed.MaxSpeed = 300;
        Decay( AUTO_DECAY );
        Mode( MODE_FULL_STEP );
    }

    if ( ( RealSpeed.MinSpeed == 0 ) || ( RealSpeed.MinSpeed == 0xffff ) )
    {
        RealSpeed.MinSpeed = 10;
    }

    if ( ( RealSpeed.Acceleration == 0 ) || ( RealSpeed.Acceleration == 0xffff ) )
    {
        RealSpeed.Acceleration = 100;
    }

    if ( ( StepsPerRev == 0 ) || ( StepsPerRev == 0xffff ) )
    {
        StepsPerRev = 400;
    }

    Disabled_Stepper( );
    Decay( CurrentDecay );
    Mode( CurrentMode );
    RealToTime( );
    CalWorkParam( );
}
```

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

mode	the first argument.

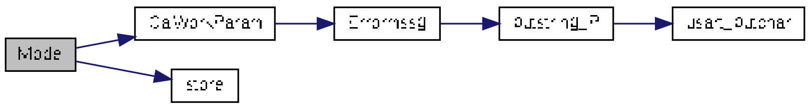
Returns

void

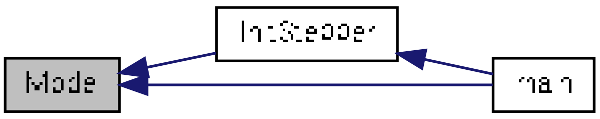
Definition at line 215 of file drv_8825.c.

```
{
    CurrentMode = mode;
    CalWorkParam( );
    store( );

    if ( mode == MODE_FULL_STEP )
    {
        clr_bit( PMODE0, MODE0 );
        clr_bit( PMODE1, MODE1 );
        clr_bit( PMODE2, MODE2 );
    }

    if ( mode == MODE_HALF_STEP )
    {
        set_bit( PMODE0, MODE0 );
        clr_bit( PMODE1, MODE1 );
        clr_bit( PMODE2, MODE2 );
    }

    if ( mode == MODE_QUATER_STEP )
    {
        clr_bit( PMODE0, MODE0 );
        set_bit( PMODE1, MODE1 );
        clr_bit( PMODE2, MODE2 );
    }

    if ( mode == MODE_8_MICROSTEP )
    {
        set_bit( PMODE0, MODE0 );
        set_bit( PMODE1, MODE1 );
        clr_bit( PMODE2, MODE2 );
    }

    if ( mode == MODE_16_MICROSTEP )
    {
        clr_bit( PMODE0, MODE0 );
        clr_bit( PMODE1, MODE1 );
        set_bit( PMODE2, MODE2 );
    }

    if ( mode == MODE_32_MICROSTEP )
    {
        set_bit( PMODE0, MODE0 );
        clr_bit( PMODE1, MODE1 );
        set_bit( PMODE2, MODE2 );
    }

}
```

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 timmer speed.

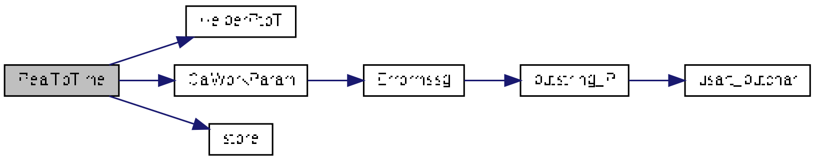
Returns

void

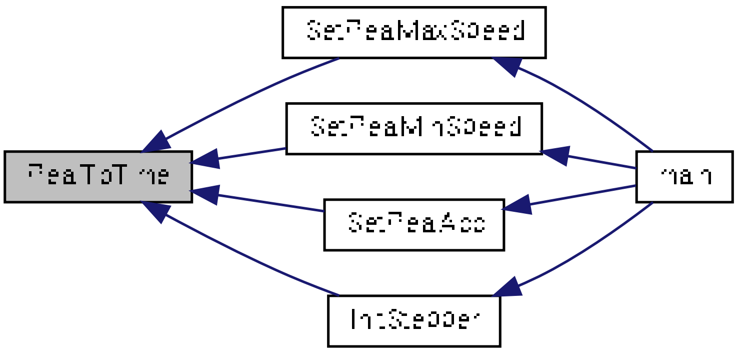
Definition at line 622 of file drv_8825.c.

```
{

    uint16_t revpermin;
    TimeParam.MaxSpeed=HelperRtoT(RealSpeed.
  MaxSpeed);
    TimeParam.MinSpeed=HelperRtoT(RealSpeed.
  MinSpeed);

    revpermin = ( TimeParam.MinSpeed - TimeParam.
  MaxSpeed ) / RealSpeed.Acceleration;

    //putstring_P( PSTR( "acc=" ) );
    //putstring( int_to_string( ( uint64_t ) ( revpermin ) ) );
    //putstring_P( PSTR( "\r" ) );

    if ( revpermin < 1 )
    {
        TimeParam.Acceleration = 1;
        //  putstring_P( PSTR( "Acc  low\r" ) );
    } else
    {
        TimeParam.Acceleration = revpermin;
    }

    CalWorkParam( );
    store( );
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.23 void Set_Steps_revol (uint16_t *step_rev*)

Sets the steps per revolution.

See Also

StepsPerRev

Parameters

step_rev	the first argument.

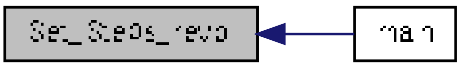
Returns

void

Definition at line 535 of file drv_8825.c.

```
{
    StepsPerRev = step_rev;
}
```

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

accel	the first argument.

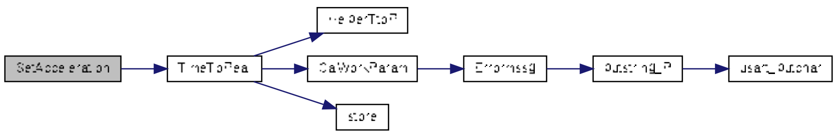
Returns

void

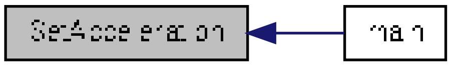
Definition at line 153 of file drv_8825.c.

```
{
    TimeParam.Acceleration = accel;
    TimeToReal( );
}
```

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

spd	the first argument.

Returns

void

Definition at line 468 of file drv_8825.c.

```
{
    Motion.CurrentSpeed = spd;
}
```

5.2.3.26 void SetMaxSpeed (uint16_t *spd*)

Function that sets the maximum speed.

See Also

TimeParam.MaxSpeed

Parameters

spd	the first argument.

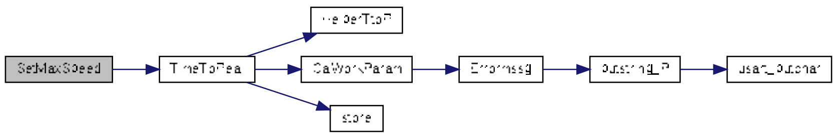
Returns

void

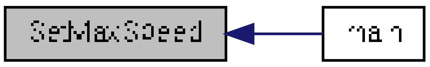
Definition at line 128 of file drv_8825.c.

```
{
    TimeParam.MaxSpeed = spd;
    TimeToReal( );
}
```

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

spd	the first argument.

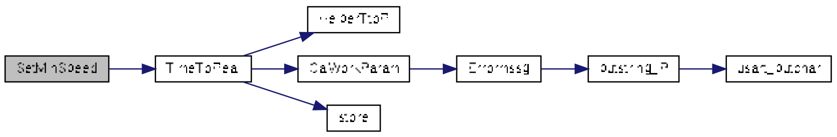
Returns

void

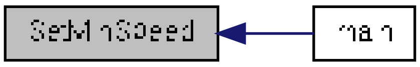
Definition at line 141 of file drv_8825.c.

```
{
    TimeParam.MinSpeed = spd;
    TimeToReal( );
}
```

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

realacc	the first argument.

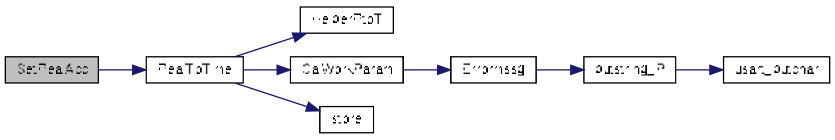
Returns

void

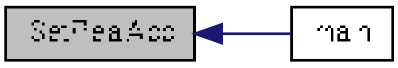
Definition at line 581 of file drv_8825.c.

```
{
    RealSpeed.Acceleration = realacc;
    RealToTime( );
}
```

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

maxspd	the first argument.

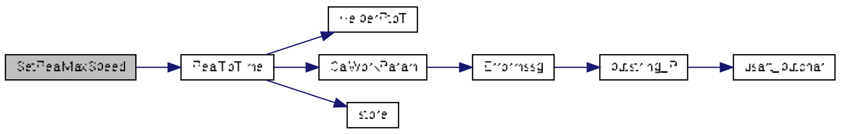
Returns

void

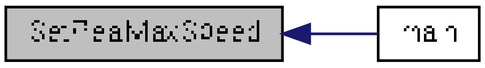
Definition at line 557 of file drv_8825.c.

```
{
    RealSpeed.MaxSpeed = maxspd;
    RealToTime( );
}
```

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

minspd	the first argument.

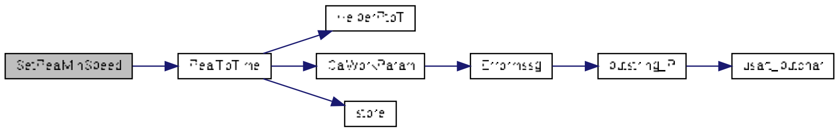
Returns

void

Definition at line 569 of file drv_8825.c.

```
{
    RealSpeed.MinSpeed = minspd;
    RealToTime( );
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

SetRealMinSpeed
main

5.2.3.31 void Step (step_type *step*)

Performs single step.

See Also

step_type

Parameters

step	the first argument.

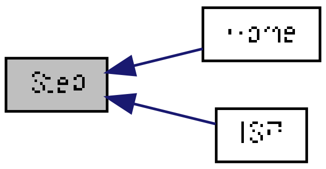
Returns

void

Definition at line 272 of file drv_8825.c.

```
{
    if ( step == STEP_CLOCKWISE )
    {
        set_bit( PDIRECTION, DIRECTION );

    } else
    {
        clr_bit( PDIRECTION, DIRECTION );
    }
    set_bit( PSTEP, STEP );
    _delay_us( 2 );
    clr_bit( PSTEP, STEP );
    _delay_us( 2 );
}
```

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.

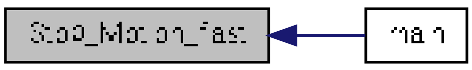
Returns

void

Definition at line 361 of file drv_8825.c.

```
{
    ConstSpd = 0;
    Motion.StepsAccel = 0;
    Motion.StepsConstSpeed = 0;
    Motion.StepsDeaccel = 0;
}
```

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.

```
{
    ConstSpd = 0;
    Motion.StepsAccel = 0;
    Motion.StepsConstSpeed = 0;
}
```

Here is the caller graph for this function:

Stop_Motion_normal
main

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.

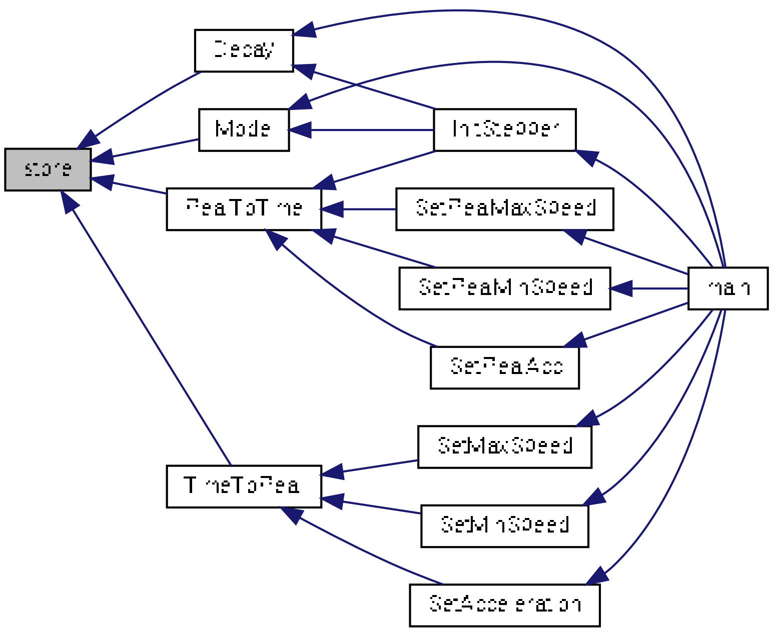
Returns

void

Definition at line 820 of file drv_8825.c.

```
{
    eeprom_write_word( ( uint16_t * ) ( 0 ), RealSpeed.MaxSpeed );
    eeprom_write_word( ( uint16_t * ) ( 2 ), RealSpeed.MinSpeed );
    eeprom_write_word( ( uint16_t * ) ( 4 ), RealSpeed.Acceleration );
    eeprom_write_word( ( uint16_t * ) ( 6 ), StepsPerRev );
    eeprom_write_byte( ( uint8_t * ) ( 8 ), CurrentMode );
    eeprom_write_byte( ( uint8_t * ) ( 9 ), CurrentDecay );


}
```

Here is the caller graph for this function:



5.2.3.35 void TimeToReal (void)

Calculates the timmer speed to the real speed.

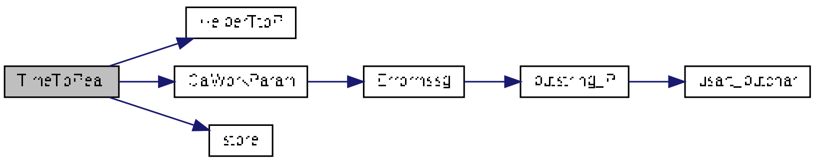
Returns

void

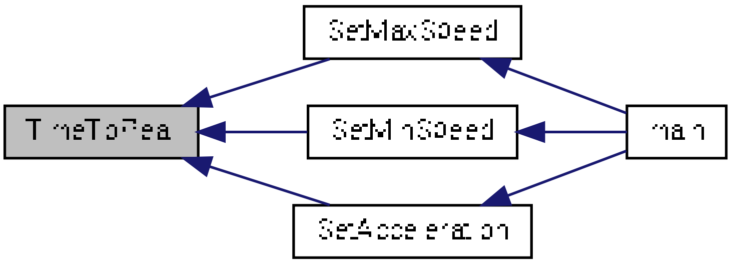
Definition at line 652 of file drv_8825.c.

```
{
    uint16_t r;
    RealSpeed.MaxSpeed=HelperTtoR(TimeParam.
  MaxSpeed);
    RealSpeed.MinSpeed=HelperTtoR(TimeParam.
  MinSpeed);

/*


    //const uint64_t l=1000000000*60;

    r = TimeParam.MaxSpeed;

    r = r * PERTMR;
    r = r*StepsPerRev;
    r = 10000000000 / r;
    r = r * 6;




    if ( r > 1 )
    {
        RealSpeed.MaxSpeed = r;
    } else
    {
        RealSpeed.MaxSpeed = 1;
        //putstring_P( PSTR( "Target MaxSpeed\r" ) );
    }


    r = TimeParam.MinSpeed;
    r = r * PERTMR;
    r = r*StepsPerRev;
    r = 60000000000 / r;
    //r=r*6;


    if ( r > 1 )
    {
        RealSpeed.MinSpeed = r;
    } else
    {
        RealSpeed.MinSpeed = 1;
        //putstring_P( PSTR( "Target Minspeed\r" ) );
    }

*/

    r = ( TimeParam.MinSpeed - TimeParam.MaxSpeed ) / TimeParam.Acceleration;

    if ( r < 1 )
    {
        RealSpeed.Acceleration = 1;
        //putstring_P( PSTR( "Acc low\r" ) );
    } else
    {
        RealSpeed.Acceleration = r;
    }

    CalWorkParam( );
    store( );
}
```

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

step	the first argument.

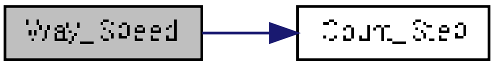
Returns

void

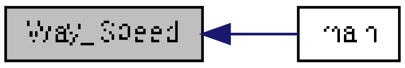
Definition at line 339 of file drv_8825.c.

```
{
    ConstSpd = 1;
    Count_Step( step, 60000 );
}
```

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 continuos 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 continuos loop for controlling it.

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

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

Note

this is whear 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

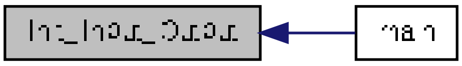
Returns

void

Definition at line 561 of file main.c.

```
{
    //input init

    set_as_input( DRXD, RXD );
    set_as_input( DHOME, HOME );
    set_as_input( DSCL, SCL );
    set_as_input( DMISO, MISO );
    set_as_input( DFAULT, FAULT );

    set_bit( PRXD, RXD );
    set_bit( PSCL, SCL );
    set_bit( PMISO, MISO );
    set_bit(PHOME, HOME);

    //output init

    set_as_output( DENABLE_STEPPER, ENABLE_STEPPER );
    set_as_output( DTXD, TXD );
    set_as_output( DMODE0, MODE0 );
    set_as_output( DMODE1, MODE1 );
    set_as_output( DMODE2, MODE2 );
    set_as_input( DDECAY, DECAY );
    set_as_output( DSTEP, STEP );
    set_as_output( DMOSI, MOSI );
    set_as_output( DDIRECTION, DIRECTION );
    set_as_output( DDIR_485, DIR_485 );

    set_bit( PENABLE_STEPPER, ENABLE_STEPPER );
    clr_bit( PMODE0, MODE0 );
    clr_bit( PMODE1, MODE1 );
    clr_bit( PMODE2, MODE2 );
    clr_bit( PDECAY, DECAY );
    clr_bit( PDIRECTION, DIRECTION );
    set_bit( PDIR_485, DIR_485 );

}
```

Here is the caller graph for this function:



5.3.2.2 void test (void)

Just for testing

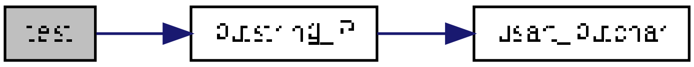
Returns

void

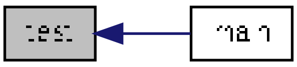
Definition at line 603 of file main.c.

```
{
    putstring_P( PSTR( "Test?" ) );

}
```

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

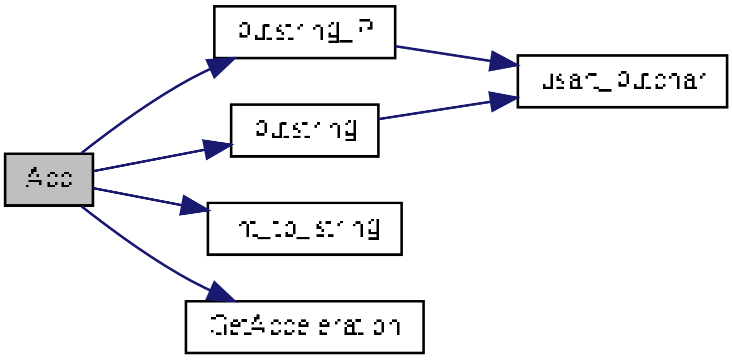
5.4.2 Function Documentation

5.4.2.1 void Acc (void)

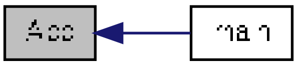
Definition at line 77 of file messages.c.

```
{
    putstring_P( PSTR( "Acc=" ) );
    putstring( int_to_string( ( uint64_t ) (
   GetAcceleration( ) ) ) );
    putstring_P( PSTR( "\r" ) );
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

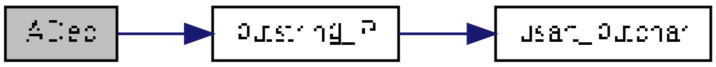


5.4.2.2 void ADec (void)

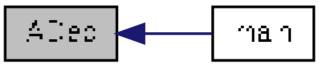
Definition at line 42 of file messages.c.

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

Here is the call graph for this function:



Here is the caller graph for this function:

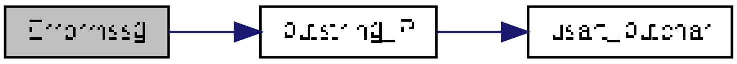


5.4.2.3 void Errormssg (void)

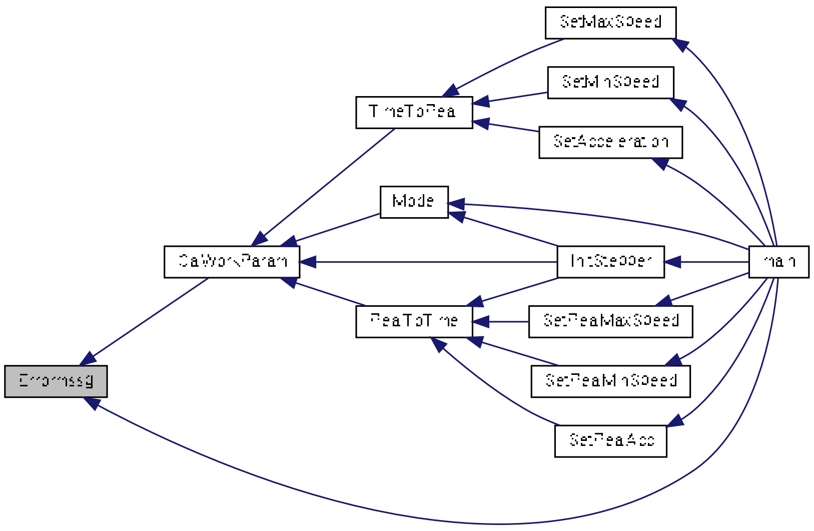
Definition at line 91 of file messages.c.

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

Here is the call graph for this function:



Here is the caller graph for this function:

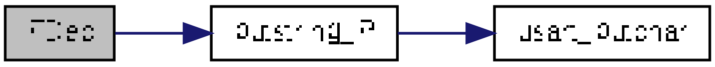


5.4.2.4 void FDec (void)

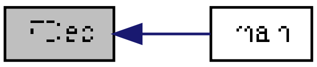
Definition at line 37 of file messages.c.

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

Here is the call graph for this function:



Here is the caller graph for this function:

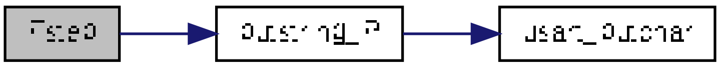


5.4.2.5 void Fstep (void)

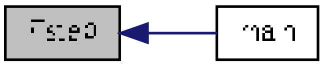
Definition at line 47 of file messages.c.

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

Here is the call graph for this function:



Here is the caller graph for this function:



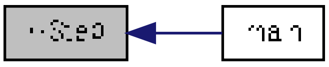
5.4.2.6 void HStep (void)

Definition at line 52 of file messages.c.

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

Here is the call graph for this function:

Here is the caller graph for this function:

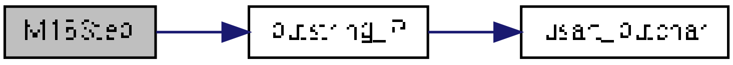


5.4.2.7 void M16Step (void)

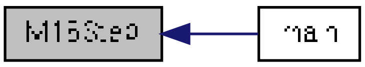
Definition at line 67 of file messages.c.

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

Here is the call graph for this function:



Here is the caller graph for this function:

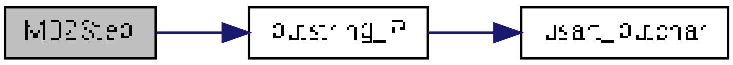


5.4.2.8 void M32Step (void)

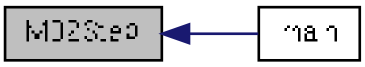
Definition at line 72 of file messages.c.

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

Here is the call graph for this function:



Here is the caller graph for this function:



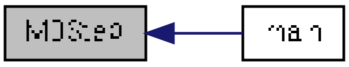
5.4.2.9 void M8Step (void)

Definition at line 62 of file messages.c.

```
{
    putstring_P( PSTR( "8 microsteps\r" ) );
}
```

Here is the call graph for this function:

Here is the caller graph for this function:

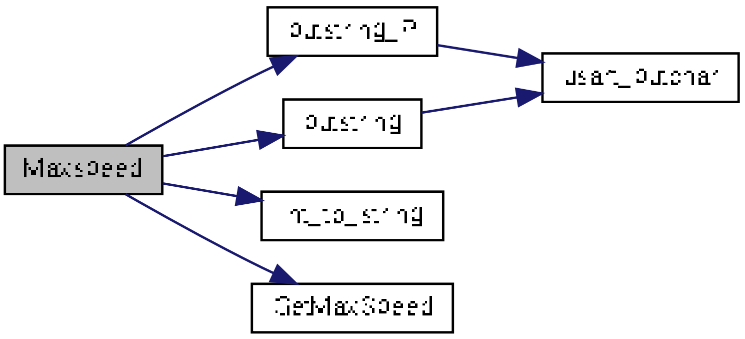


5.4.2.10 void Maxspeed (void)

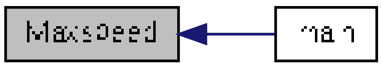
Definition at line 96 of file messages.c.

```
{
    putstring_P( PSTR( "MaxSPD=" ) );
    putstring( int_to_string( ( uint64_t ) ( GetMaxSpeed( ) ) ) );
     putstring_P( PSTR( "\r" ) );
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

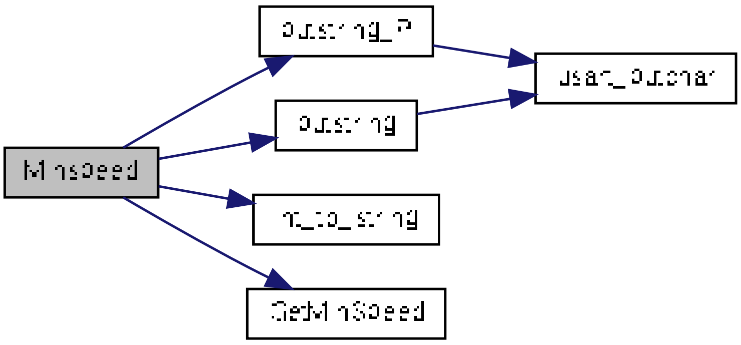


5.4.2.11 void Minspeed (void)

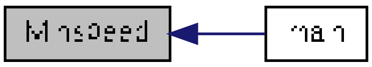
Definition at line 84 of file messages.c.

```
{
    putstring_P( PSTR( "MinSPD=" ) );
    putstring( int_to_string( ( uint64_t ) ( GetMinSpeed( ) ) ) );
    putstring_P( PSTR( "\r" ) );
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

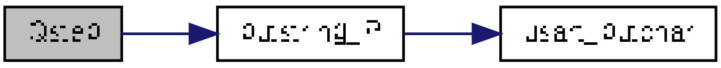


5.4.2.12 void Qstep (void)

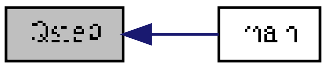
Definition at line 57 of file messages.c.

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

Here is the call graph for this function:



Here is the caller graph for this function:

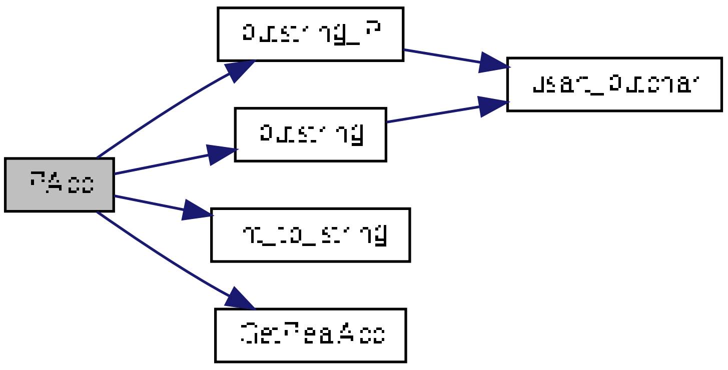


5.4.2.13 void RAcc (void)

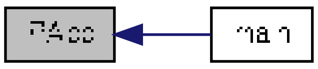
Definition at line 124 of file messages.c.

```
{
    putstring_P( PSTR( "RAcc=" ) );
    putstring( int_to_string( ( uint16_t ) ( GetRealAcc( ) ) ) );
    putstring_P( PSTR( "\r" ) );
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

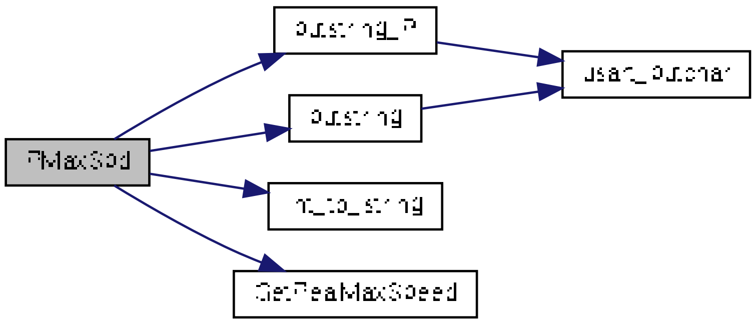


5.4.2.14 void RMaxSpd (void)

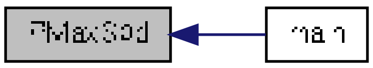
Definition at line 117 of file messages.c.

```
{
    putstring_P( PSTR( "RMaxSPD=" ) );
    putstring( int_to_string( ( uint64_t ) (
  GetRealMaxSpeed( ) ) ) );
    putstring_P( PSTR( "\r" ) );
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

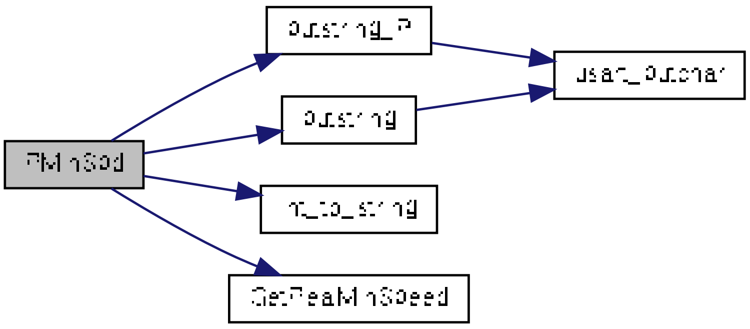


5.4.2.15 void RMinSpd (void)

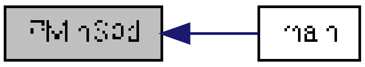
Definition at line 110 of file messages.c.

```
{
    putstring_P( PSTR( "RMinSpd=" ) );
    putstring( int_to_string( ( uint16_t ) (
  GetRealMinSpeed( ) ) ) );
    putstring_P( PSTR( "\r" ) );
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

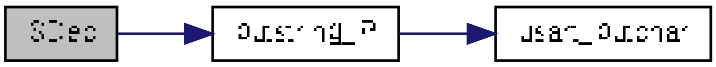


5.4.2.16 void SDec (void)

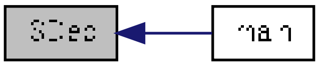
Definition at line 32 of file messages.c.

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

Here is the call graph for this function:



Here is the caller graph for this function:

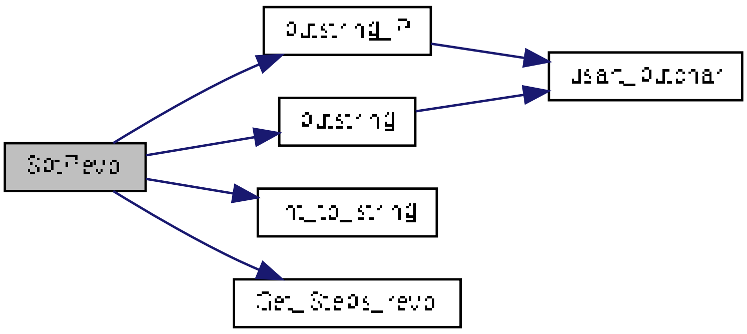


5.4.2.17 void SptRevol (void)

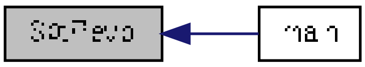
Definition at line 103 of file messages.c.

```
{
    putstring_P( PSTR( "SPR=" ) );
    putstring( int_to_string( ( uint16_t ) (
  Get_Steps_revol( ) ) ) );
    putstring_P( PSTR( "\r" ) );
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

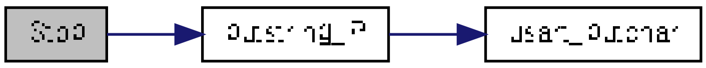


5.4.2.18 void Stop (void)

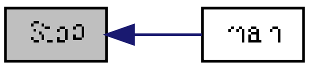
Definition at line 131 of file messages.c.

```
{
    putstring_P( PSTR( "Stop" ) );
}
```

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 UART_RX_BUFFER_SIZE and UART_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_RX_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

xtalcpu	system clock in Mhz, e.g. 4000000L for 4Mhz
baudrate	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

xtalcpu	system clock in Mhz, e.g. 4000000L for 4Mhz
baudrate	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

void	

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

FILE	stream

Returns

byte from ring buffer

Definition at line 128 of file usart.c.

```
{
    uint8_t c;

    if (usart_rx.fillcount == 0)
        return EOF;

    if (usart_rx.nlines == 0)
        return EOF;

    c = usart_rx.data[usart_rx.tail];
    usart_rx.tail = (usart_rx.tail + 1) & (usart_rx.
  size - 1);

    //Disable RXC interrupt to run next lines atomic
    UCSR0B &= ~(1 << RXCIE0);
    usart_rx.fillcount--;

    //Behavior similar to Unix stty ICRNL
    if (c == VEOL)
    {
        c = '\n';
        usart_rx.nlines--;
    }

    //Enable RXC interrupt
    UCSR0B |= (1 << RXCIE0);

    return c;
}/* usart_getchar */
```

5.5.3.2 void usart_init (unsigned int *baudrate*)

Initialize USART and set baudrate.

Parameters

baudrate	Specify baudrate using macro UART_BAUD_SELECT()

Returns

none

Initialize USART and set baudrate Baudrate using macro USART_BAUD_SELECT()

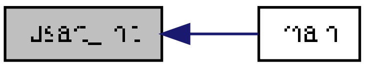
Parameters

baudrate	

Definition at line 69 of file usart.c.

```
{
    ATOMIC_BLOCK(ATOMIC_RESTORESTATE)
    {
        usart_rx.data = brx;
        usart_rx.size = BSIZE;
        usart_rx.head = 0;
        usart_rx.tail = 0;
        usart_rx.fillcount = 0;
        usart_rx.nlines = 0;

        usart_tx.data = btx;
        usart_tx.size = BSIZE;
        usart_tx.head = 0;
        usart_tx.tail = 0;
        usart_tx.fillcount = 0;
        usart_tx.nlines = 0;

        /* Set baud rate */
        if (baudrate & 0x8000)
        {
            UCSR0A = (1 << U2X0); //Enable 2x speed
            baudrate &= ~0x8000;
        }
        UBRR0H = (unsigned char) (baudrate >> 8);
        UBRR0L = (unsigned char) baudrate;

#if (USART0_TYPE) == 485

#if  defined(__AVR_ATmega8__)  || defined(__AVR_ATmega16__) || defined(__AVR_ATmega32__) \
  || defined(__AVR_ATmega323__)
        PORTD |= (1 << PD0); //pull-up RX0
#elif defined(__AVR_ATmega162__)
        PORTB |= (1 << PB2); //pull-up RX1
#else
#error "RX port is not internally pulled up"
#endif

        //Receiver output enable
        transmitter0_disable();
        set_as_output(DDRB, transceiver0);

        //Enable USART0 receiver and transmitter, receive complete
        //and transmit complete interrupt.
        UCSR0B = (1 << RXCIE0) | (1 << TXCIE0) | (1 << RXEN0) | (1 << TXEN0);
#else
        /* Enable USART receiver and transmitter and receive complete interrupt */
        UCSR0B = (1 << RXCIE0) | (1 << RXEN0) | (1 << TXEN0);
#endif
        /* Set frame format: asynchronous, 8data, no parity, 1stop bit */
        UCSR0C = (1 << URSEL0) | (1 << UCSZ01) | (1 << UCSZ00);
    }
}/* 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

data	byte to be transmitted

Returns

none

Write byte to ring buffer for transmitting via USART

Parameters

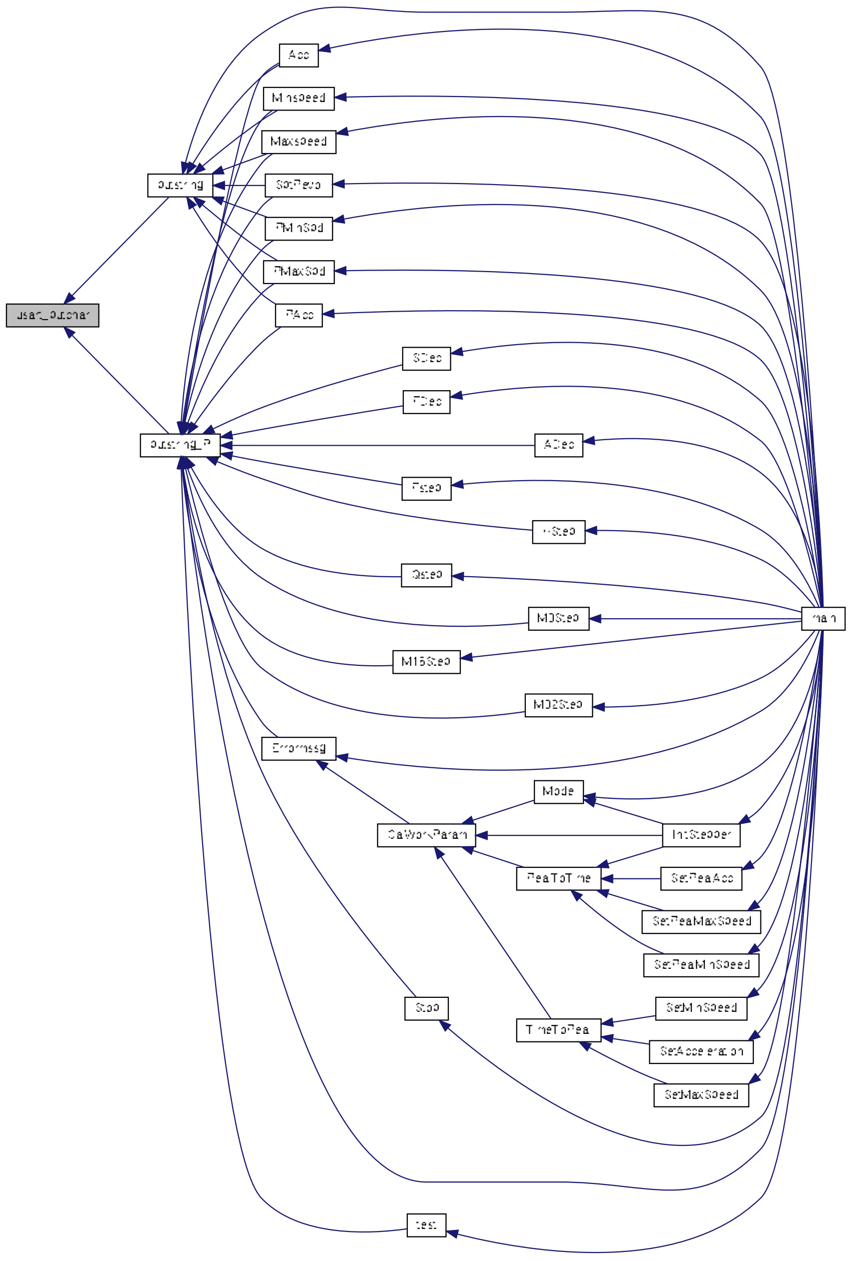
c	byte to be transmitted
stream	

Returns

Definition at line 164 of file usart.c.

```
{
#if (USART0_TYPE) == 485

    //Driver output enable
    transmitter0_enable();
#endif

    if (c == '\n')
        c = VEOL;

    while (usart_tx.fillcount == usart_tx.size)
        ; //wait for free space in buffer

    usart_tx.data[usart_tx.head] = c;
    usart_tx.head = (usart_tx.head + 1) & (usart_tx.
  size - 1);

    //Disable UDRE interrupt to run next lines atomic
    UCSR0B &= ~(1 << UDRIE0);

    usart_tx.fillcount++;

    //Enable UDRE interrupt
    UCSR0B |= (1 << UDRIE0);

    return 0;
}/* usart_putchar */
```

Here is the caller graph for this function:



5.6 utils Library

This module contains functions that are converting ASCI 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 ASCI to uint64_t.

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

This module contains functions that are converting ASCI 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 ASCI to uint64_t

Parameters

in	the first argument.

Returns

uint64_t

Definition at line 45 of file utils.c.

```
{
    char *b;
    uint64_t c=0;
    uint64_t multi=1;

    b = in;

    while ( ( (*b > 47) && (*b < 58 )) || ( *b == 32 ) || ( *b == 46 ) || ( *b ==44 ) )
    {
        b++;
    }

    while(b>=in)
    {
        if(( (*b > 47) && (*b < 58 )))
        {
            c=c+multi*(*b-'0');
            multi=multi*10;
        }

        b--;
    }
    return c;
}
```

Here is the caller graph for this function:

5.6.2.2 char∗ int_to_string (uint64_t *i*)

Function that converts intiger to string.

Parameters

t	the first argument.

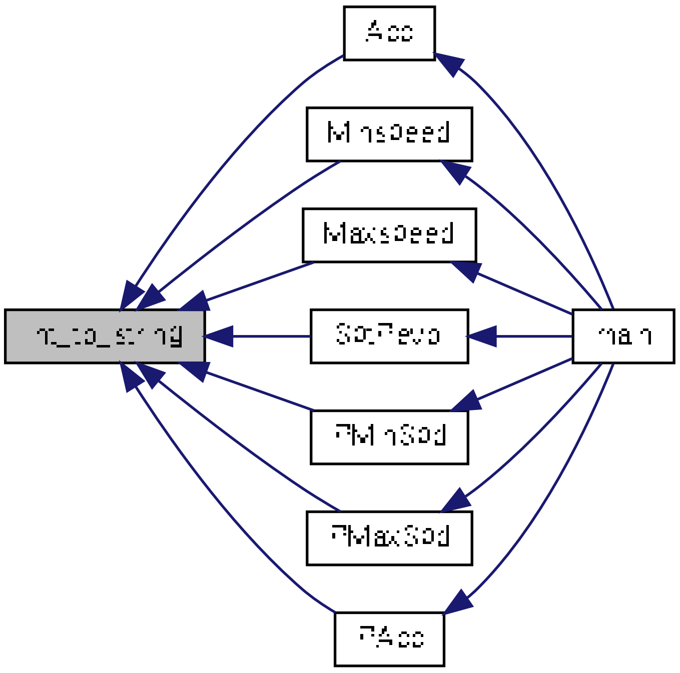
Returns

char

Definition at line 76 of file utils.c.

```
{
    unsigned char temp;
    unsigned char s = 0,t = 0;
    if(i==0)
    {
        buf[0]='0';
        buf[1]=0;
        return buf;
    }
    while(i) {
        buf[s++] = i % 10 + '0';
        i /= (uint64_t)(10);
    }
    buf[s] = 0;
    s-=1;
    for(;t<s;t++,s--) {
        temp = buf[s];
        buf[s]=buf[t];
        buf[t] = temp;
    }
    return buf;
}
```

Here is the caller graph for this function:



5.6.2.3 void putstring (const char ∗ *putc*)

Function that reaplaces the printf for uint64_t.

Parameters

putc	the first argument.

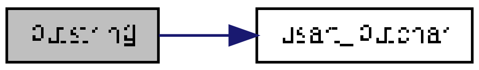
Returns

void

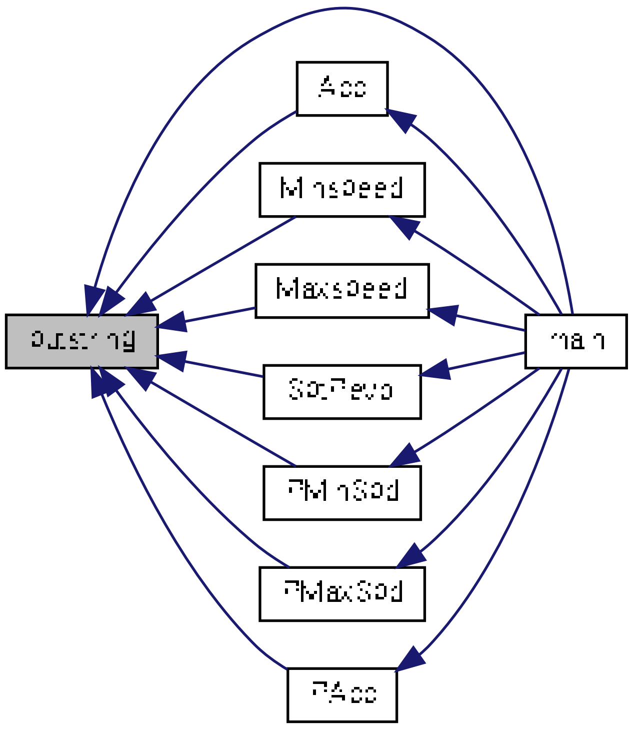
Definition at line 105 of file utils.c.

```
{
    const char *p;
    p=putc;
    while (*p!=0)
    {
        usart_putchar(*p, stdout);
        p++;
    }

    return;
}
```

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 reaplaces the printf from Programme memmory space.

Parameters

putc	the first argument.

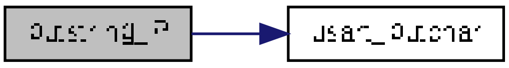
Returns

void

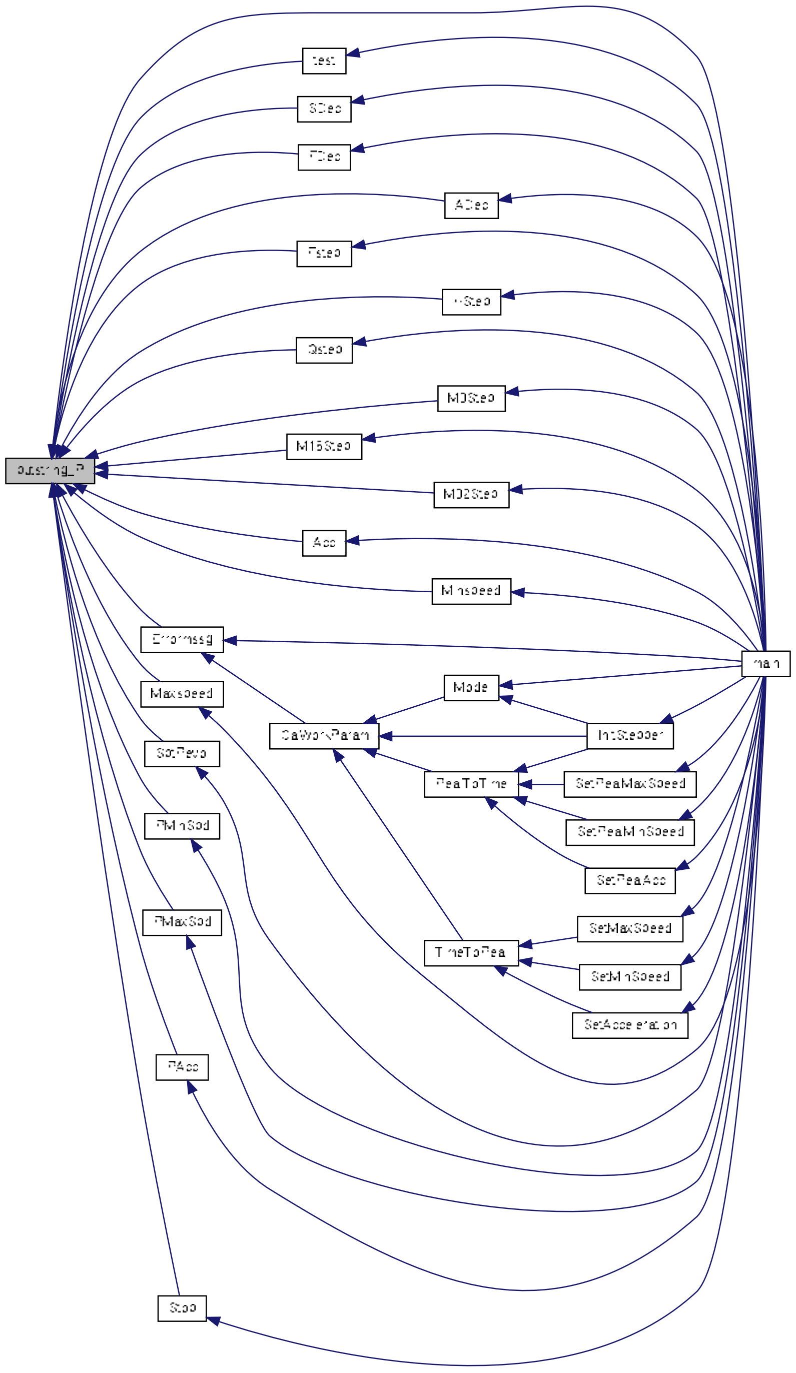
Definition at line 124 of file utils.c.

```
{
    while (pgm_read_byte ( putc )!=0)
    {
        usart_putchar(pgm_read_byte ( putc ), stdout);
        putc++;
    }

    return;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

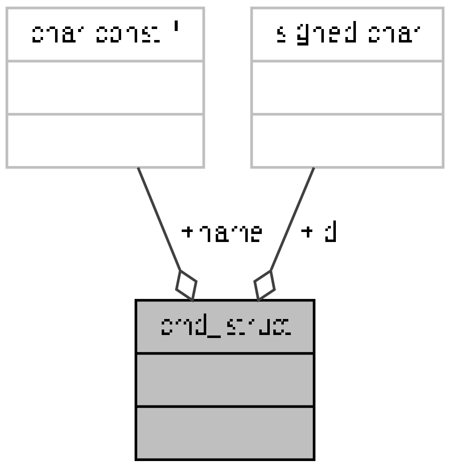


Chapter 6

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.

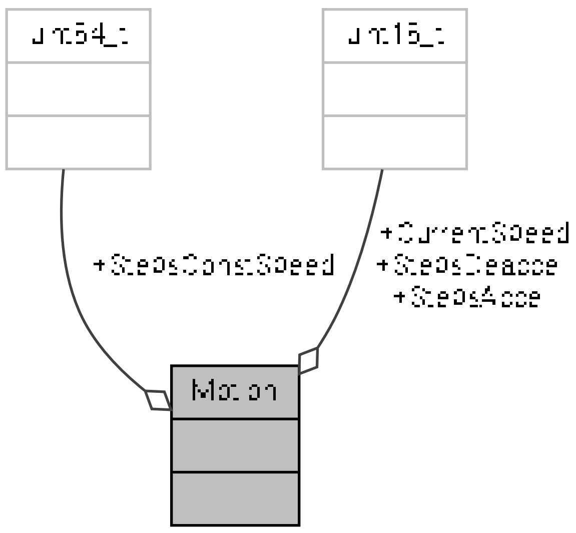
The documentation for this struct was generated from the following file:

- interpreter.c

6.2 Motion Struct Reference

This structure defines the current step execution status. This structure is used for the timmer 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 timmer 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 timmer

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.

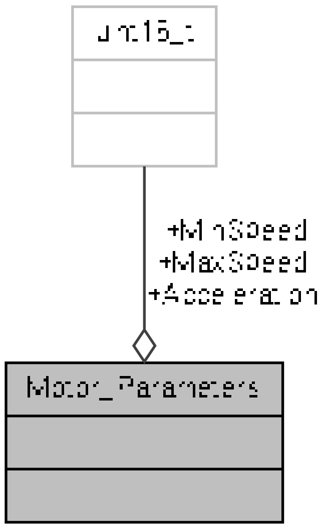
The documentation for this struct was generated from the following file:

- 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 paramenters used for timmer, 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 paramenters used for timmer, 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

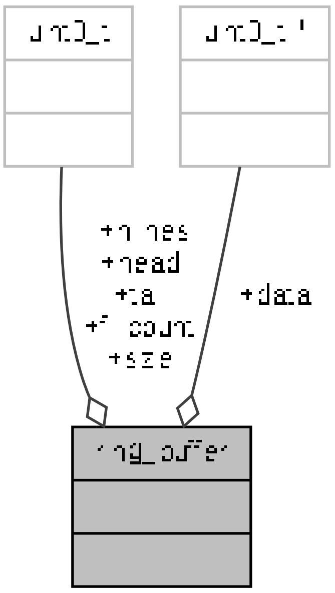
Definition at line 55 of file drv_8825.c.

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

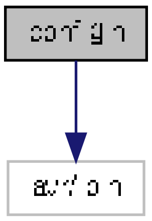
Chapter 7

File Documentation

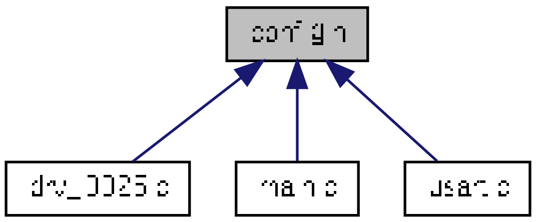
7.1 config.h File Reference

`#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))

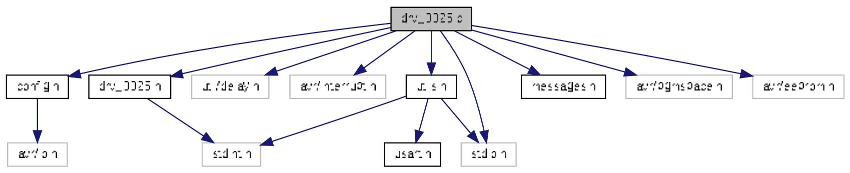
- #define CHECKPIN(x, y) ((x&(1<<y))!=0?1:0)
- #define clr_bit(byte, bit) byte &= ∼(1 << bit)
- #define clr_port(PORTx, PORTxn) PORTx &= ∼(1 << PORTxn)
- #define DDECAY DDRC
- #define DDIR_485 DDRB
- #define DDIRECTION DDRB
- #define DECAY PC1
- #define DENABLE_STEPPER DDRD
- #define DFAULT DDRB
- #define 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 timmer 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 paramenters used for timmer, 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

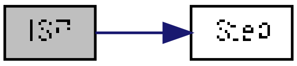
7.2.3.1 ISR (TIMER1_OVF_vect)

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

Definition at line 477 of file drv_8825.c.

```
{
    if ( ConstSpd == 1 )
    {
        Motion.StepsConstSpeed++;
    }
    if ( Motion.StepsAccel > 0 )
    {
        if ( Motion.CurrentSpeed > WorkingParam.
  Acceleration )
        {
            Motion.CurrentSpeed = Motion.CurrentSpeed -
  WorkingParam.Acceleration;
        } else
        {
            Motion.CurrentSpeed = WorkingParam.
  MaxSpeed;
        }

        if ( Motion.CurrentSpeed < WorkingParam.
  MaxSpeed )
        {
            Motion.CurrentSpeed = WorkingParam.
  MaxSpeed;
        }
        TCNT1 = 0 - Motion.CurrentSpeed;
        Step( StepDir );
        Motion.StepsAccel--;

        return;
    }

    if ( Motion.StepsConstSpeed > 0 )
    {
        Motion.StepsConstSpeed--;
        TCNT1 = 0 - WorkingParam.MaxSpeed;
        Step( StepDir );

        return;
    }


    if ( Motion.StepsDeaccel > 0 )
    {
        Motion.StepsDeaccel--;
        Motion.CurrentSpeed = Motion.CurrentSpeed +
  WorkingParam.Acceleration;
        TCNT1 = 0 - Motion.CurrentSpeed;
        Step( StepDir );

        return;
    }

    clr_bit( TIMSK, TOIE1 );
    TCNT1 = 65535 - WorkingParam.MinSpeed;
}
```

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]`

Flog 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]`

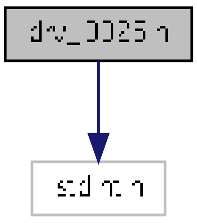
This is used for the timmer and it is calculated based on the mode.

Definition at line 89 of file drv_8825.c.

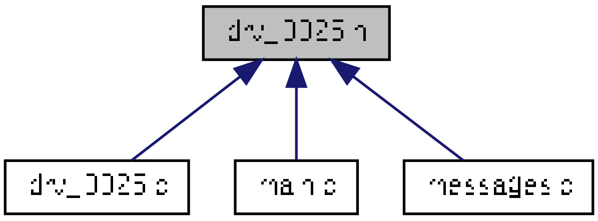
7.3 drv_8825.h File Reference

```
#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

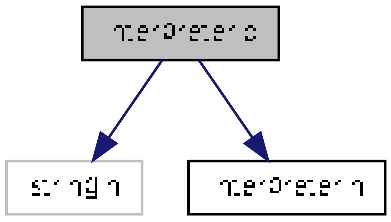
- void CalWorkParam (void)
- void Count_Step (step_type step, uint64_t step_count)
- void Decay (decay_type decay)
- void Disabled_Stepper (void)
- void Enable_Stepper (void)
- decay_type Get_Decay (void)
- mode_type Get_Mode (void)
- uint8_t Get_Start (void)
- uint16_t Get_Steps_revol (void)
- uint16_t 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

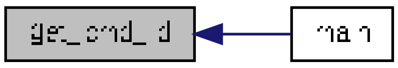
name	

Returns

Definition at line 60 of file interpreter.c.

```
{
    unsigned char i = 0;

    if (name[0] == 0)
        return -2;

    for (i = 0; cmd_tbl[i].id != -1; i++)
        if (!strncmp(name, cmd_tbl[i].name,strlen(cmd_tbl[i].name)))
            return cmd_tbl[i].id;

    return -1;
}
```

Here is the caller graph for this function:



7.4.1.2 char const∗ get_cmd_name (char *id*)

Parameters

id	

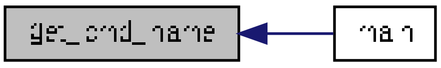
Returns

command name or NULL if not found

Definition at line 79 of file interpreter.c.

```
{
    unsigned char i;

    for (i = 0; cmd_tbl[i].id != -1; i++)
        if (id == cmd_tbl[i].id)
            return cmd_tbl[i].name;

    return NULL;
}
```

Here is the caller graph for this function:

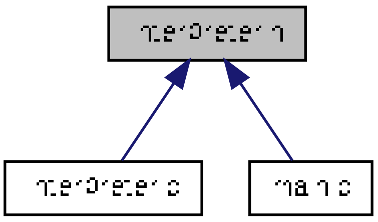


7.4.2 Variable Documentation

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,
    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

};
```

7.5.2 Function Documentation

7.5.2.1 char get_cmd_id (char * *name*)

Parameters

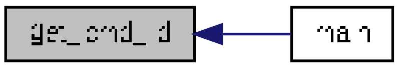
name	

Returns

Definition at line 60 of file interpreter.c.

```
{
    unsigned char i = 0;

    if (name[0] == 0)
        return -2;

    for (i = 0; cmd_tbl[i].id != -1; i++)
        if (!strncmp(name, cmd_tbl[i].name,strlen(cmd_tbl[i].name)))
            return cmd_tbl[i].id;

    return -1;
}
```

Here is the caller graph for this function:



7.5.2.2 char const∗ get_cmd_name (char *id*)

Parameters

id	

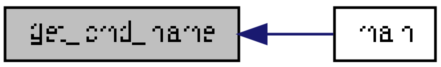
Returns

command name or NULL if not found

Definition at line 79 of file interpreter.c.

```
{
    unsigned char i;

    for (i = 0; cmd_tbl[i].id != -1; i++)
        if (id == cmd_tbl[i].id)
            return cmd_tbl[i].name;

    return NULL;
}
```

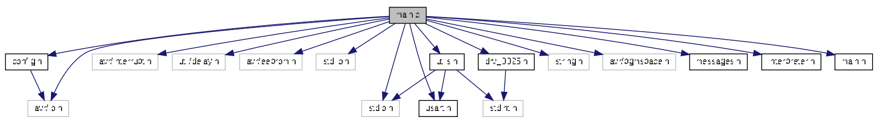
Here is the caller graph for this function:



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.

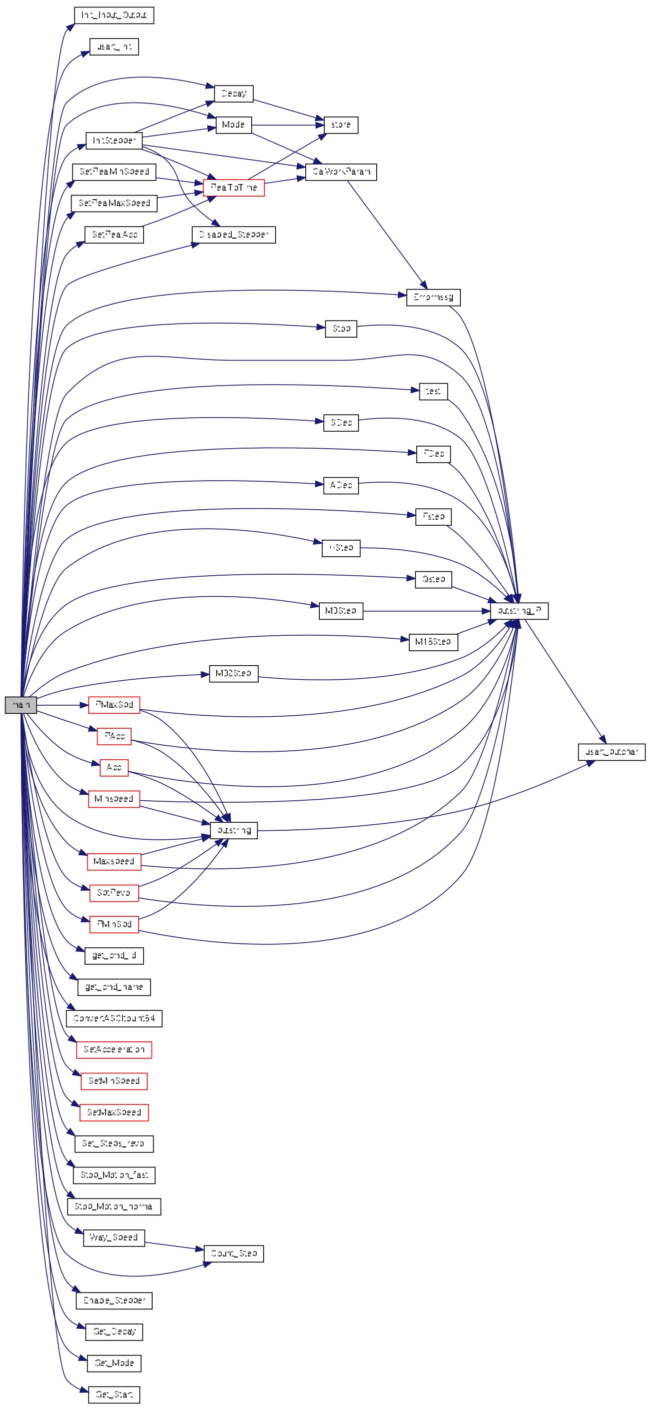
```
{
    char message[32]; // 1 byte + for string termination

    int8_t cmd;
    const char *cptr;
    char * pch;
    uint64_t ch;
    uint8_t i;


    SFIOR &= 0xfb; //11111011 -> PUD=0
    /*
     * Set ports data direction

     */
    Init_Input_Output( );

    usart_init( USART_BAUD_SELECT( USART_BAUD_RATE,
   F_CPU ) );



```

```
    /*
     * now enable interrupt, since USART library is interrupt controlled

     */
    sei( );
    stdout = stdin = &usart_str;

    _delay_ms( 100 );


    InitStepper();


    for (;; )
    {

        if ( ( cptr = fgets( message, sizeof message - 1, stdin ) ) )
        {


             cmd = get_cmd_id( message );

             switch ( cmd )
             {
                 case VERSION:
                     putstring_P( PSTR( "1.0\r" ) );
                     break;


                 case HELP:
                     //  putstring_P( PSTR( "help?\r" ) );

                     i = 0;

                     do
                     {
                         cptr = get_cmd_name( i++ );
                         if ( cptr )
                         {
                             putstring( cptr );
                             putstring_P( PSTR( "\r" ) );
                         }
                     }
                     while ( cptr );

                     break;

                 case TEST:
                     test( );
                     break;

                 case SET_DEC_SLOW:
                     Decay( SLOW_DECAY );
                     SDec( );
                     break;

                 case SET_DEC_FAST:
                     Decay( FAST_DECAY );
                     FDec( );
                     break;

                 case SET_DEC_AUTO:
                     Decay( AUTO_DECAY );
                     ADec( );
                     break;

                 case SET_MODE_FULL:
                     Mode( MODE_FULL_STEP );
                     Fstep( );
                     break;

                 case SET_MODE_HALF:
                     Mode( MODE_HALF_STEP );
                     HStep( );
                     break;

                 case SET_MODE_QUATER:
                     Mode( MODE_QUATER_STEP );
                     Qstep( );
                     break;

                 case SET_MODE_8:
                     Mode( MODE_8_MICROSTEP );
                     M8Step( );
                     break;

                 case SET_MODE_16:
                     Mode( MODE_16_MICROSTEP );
                     M16Step( );
                     break;

                 case SET_MODE_32:
                     Mode( MODE_32_MICROSTEP );
                     M32Step( );
                     break;

                 case SET_HOME:
                     //              putstring("home%d\r", Home());
                     putstring_P( PSTR( "home " ) );
                     if ( !(check_pin( IHOME, HOME )) )
                     {
                         putstring_P( PSTR( "TRUE" ) );
                     } else
                     {
                         putstring_P( PSTR( "FALSE" ) );
                     }
                     //              putstring(ConvertASCItouint64(Home()));
                     putstring_P( PSTR( "\r" ) );

                     break;

                 case SET_ACC:
                     pch = strstr( message, "=" );
                     if ( pch != NULL )
                     {
                         pch = strstr( pch, "=" );
                         ch = ConvertASCItouint64( pch + 1 );
                         if ( ch > 0 )
                         {
                             SetAcceleration( ch );
                             Acc( );
                             break;
                         } else
                         {
                             Errormssg( );
                             break;
                         }
                     }
                     break;

                 case SET_MIN_SPEED:
                     pch = strstr( message, "=" );
                     if ( pch != NULL )
                     {
                         pch = strstr( pch, "=" );
                         ch = ConvertASCItouint64( pch + 1 );
                         if ( ch > 0 )
                         {
                             SetMinSpeed( ch );
                             Minspeed( );
                             break;
                         } else
                         {
                             Errormssg( );
                             break;
                         }
                     }
                     break;

                 case SET_MAX_SPEED:
                     pch = strstr( message, "=" );
                     if ( pch != NULL )
                     {
                         pch = strstr( pch, "=" );
                         ch = ConvertASCItouint64( pch + 1 );
                         if ( ch > 0 )
                         {
                             SetMaxSpeed( ch );
                             Maxspeed( );
                             break;
                         } else
                         {
                             Errormssg( );
                             break;
                         }
                     }
                     break;

                 case SET_STP_REV:

                     pch = strstr( message, "=" );
                     if ( pch != NULL )
                     {
                         pch = strstr( pch, "=" );
                         ch = ConvertASCItouint64( pch + 1 );
                         if ( ch > 0 )
                         {
                             Set_Steps_revol( ch );
                             SptRevol();
                             break;
                         }
                         else
                         {
                             Errormssg( );
                             break;
                         }
                     }

                     break;

                 case SET_R_MIN_SPD:

                     pch = strstr( message, "=" );
                     if ( pch != NULL )
                     {
                         //pch = strstr( pch, "=" );
                         ch = ConvertASCItouint64( pch + 1 );
                         if ( ch > 0 )
                         {
                             SetRealMinSpeed( ch );
                             RMinSpd();
                             break;
                         }
                         else
                         {
                             Errormssg( );
                             break;
                         }
                     }

                     break;

                 case SET_R_MAX_SPD:

                     pch = strstr( message, "=" );
                     if ( pch != NULL )
                     {
                         pch = strstr( pch, "=" );
                         ch = ConvertASCItouint64( pch + 1 );
                         if ( ch > 0 )
                         {


                             SetRealMaxSpeed( ch );
                             RMaxSpd();
                             break;
                         }
                         else
                         {
                             Errormssg( );
                             break;
                         }
                     }

                     break;


                 case SET_R_ACC:

                     pch = strstr( message, "=" );
                     if ( pch != NULL )
                     {
                         pch = strstr( pch, "=" );
                         ch = ConvertASCItouint64( pch + 1 );
                         if ( ch > 0 )
                         {
                             SetRealAcc( ch );
                             RAcc();
                             break;
                         } else
                         {
                             Errormssg( );
                             break;
                         }
                     }

                     break;


                 case GET_ACC:
                     Acc( );
```

```
                    break;

                case GET_MIN_SPEED:
                    Minspeed( );
                    break;

                case GET_MAX_SPEED:
                    Maxspeed( );
                    break;


                case GET_STP_REV:
                    SptRevol();
                    break;


                case GET_R_MIN_SPD:
                    RMinSpd();
                    break;

                case GET_R_MAX_SPD:
                    RMaxSpd();
                    break;

                case GET_R_ACC:
                    RAcc();
                    break;



                case STOPF:
                    Stop_Motion_fast();
                    Stop();
                    break;

                case STOPN:
                    Stop_Motion_normal();
                    Stop();
                    break;

                case CONSTSPD:
                    pch = strstr( message, "clock" );
                    if ( pch != NULL )
                    {
                        Way_Speed(STEP_CLOCKWISE);
                            putstring_P( PSTR( "DONE\r " ) );
                            break;

                    }

                    pch = strstr( message, "against" );
                    if ( pch != NULL )
                    {

                            Way_Speed( STEP_COUNTER_CLOCKWISE );
                            putstring_P( PSTR( "DONE\r" ) );
                            break;
                    }

                    Errormssg();
                    break;


                case ENABLE:
                    Enable_Stepper( );
                    //putstring_P( PSTR( "Enable\r" ) );
                    break;

                case DISABLE:
                    Disabled_Stepper( );
                    //putstring_P( PSTR( "Disable\r" ) );
                    break;





                case GOTO:
                    pch = strstr( message, "clock" );
                    if ( pch != NULL )
                    {
                        pch = strstr( pch, " " );
                        ch = ConvertASCItouint64( pch );

                        if ( ch > 0 )
                        {
                            Count_Step( STEP_CLOCKWISE, ch );
```

```
/*
                            putstring_P( PSTR( "DONE " ) );

                            putstring( int_to_string( ch ) );

                            putstring_P( PSTR( " steps\r" ) );

*/
                            break;
                        } else
                        {
                            Errormssg( );
                            break;
                        }
                    }

                    pch = strstr( message, "against" );
                    if ( pch != NULL )
                    {
                        pch = strstr( pch, " " );
                        ch = ConvertASCItouint64( pch );
                        if ( ch > 0 )
                        {
                            Count_Step( STEP_COUNTER_CLOCKWISE, ch );
/*
                            putstring_P( PSTR( "DONE " ) );

                            putstring( int_to_string( ch ) );

                            putstring_P( PSTR( " steps\r" ) );

*/
                            break;
                        } else
                        {
                            Errormssg( );
                            break;
                        }
                    }
                    break;

                case STATUS:
                    switch ( Get_Decay( ) )
                    {
                        case SLOW_DECAY:
                            SDec( );
                            break;

                        case FAST_DECAY:
                            FDec( );
                            break;

                        case AUTO_DECAY:
                            ADec( );
                            break;

                        default:
                        {
                            //putstring_P( PSTR( "Not set decay\r" ) );
                            break;
                        }
                    }

                    switch ( Get_Mode( ) )
                    {
                        case MODE_FULL_STEP:
                            Fstep( );
                            break;

                        case MODE_HALF_STEP:
                            HStep( );
                            break;

                        case MODE_QUATER_STEP:
                            Qstep( );
                            break;

                        case MODE_8_MICROSTEP:
                            M8Step( );
                            break;

                        case MODE_16_MICROSTEP:
                            M16Step( );
                            break;

                        case MODE_32_MICROSTEP:
                            M32Step( );
                            break;

                        default:
                        {
                            //putstring_P( PSTR( "Not set mode\r" ) );
                            break;
                        }

                    }


                    if ( Get_Start( ) )
                    {
                        //putstring_P( PSTR( "Enable\r" ) );
                    }
                    else
                    {
                        //putstring_P( PSTR( "Disable\r" ) );
                    }

                    break;







                default:

                    if ( ( cmd < 0 ) && ( !( ( message[0] == 0x0d ) || ( message[0] == 0x0a ) ) ) )
                        putstring_P( PSTR( "Comand not implement\r" ) );
                    break;

            } //switch (cmd)
        } //if (USART_RxEOL > 0)




    } //for (;;)

    return 0;
} //End int main(void)
```

Here is the call graph for this function:



7.6.3 Variable Documentation

7.6.3.1 FILE usart_str

Initial value:

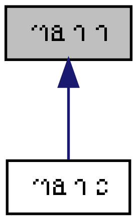
```
=
    FDEV_SETUP_STREAM( usart_putchar, usart_getchar, _FDEV_SETUP_RW )
```

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:



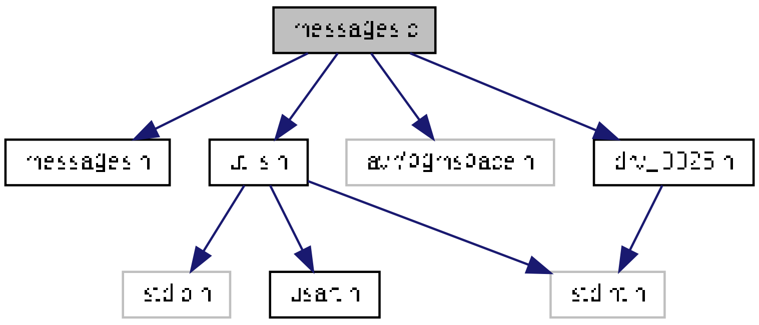
Functions

- 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:

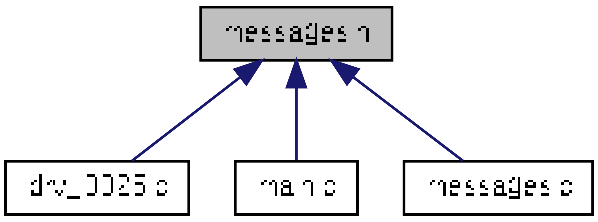


Functions

- 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

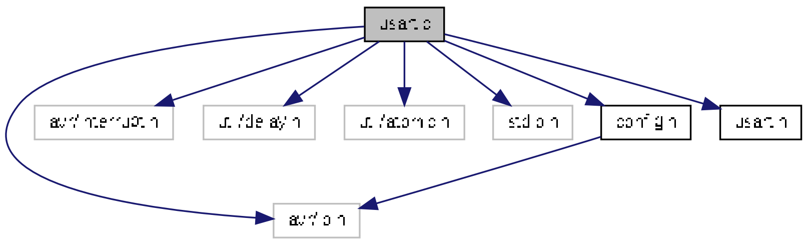
- void Acc (void)
- void ADec (void)
- void Errormssg (void)
- void FDec (void)
- void Fstep (void)
- void HStep (void)
- void M16Step (void)
- void M32Step (void)
- void M8Step (void)
- void Maxspeed (void)
- void Minspeed (void)

- 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.

```
{
    uint8_t data;

    data = UDR0;

    if (usart_rx.fillcount < usart_rx.size)
    {
        if (data == VEOL)
            usart_rx.nlines++;

        usart_rx.data[usart_rx.head] = data;

        usart_rx.head = (usart_rx.head + 1) & (usart_rx.size - 1);
        usart_rx.fillcount++;
    }
    else
    {
        //The Buffer is full! clear it
        usart_rx.head = 0;
        usart_rx.tail = 0;
        usart_rx.fillcount = 0;
        usart_rx.nlines = 0;

        /* Wait for empty transmit buffer */
        while (!(UCSR0A & (1 << UDRE0)))
            ;
        UDR0 = 'B';
        while (!(UCSR0A & (1 << UDRE0)))
            ;
        UDR0 = 'O';
        while (!(UCSR0A & (1 << UDRE0)))
```

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

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.

```
{
    if (usart_tx.fillcount > 0)
    {
        UDR0 = usart_tx.data[usart_tx.tail];

        usart_tx.tail = (usart_tx.tail + 1) & (usart_tx.
   size - 1);
        usart_tx.fillcount--;
    }
    else
    {
        /* tx buffer empty, disable UDRE interrupt */
        UCSR0B &= ~(1 << UDRIE0);
    }
}
```

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

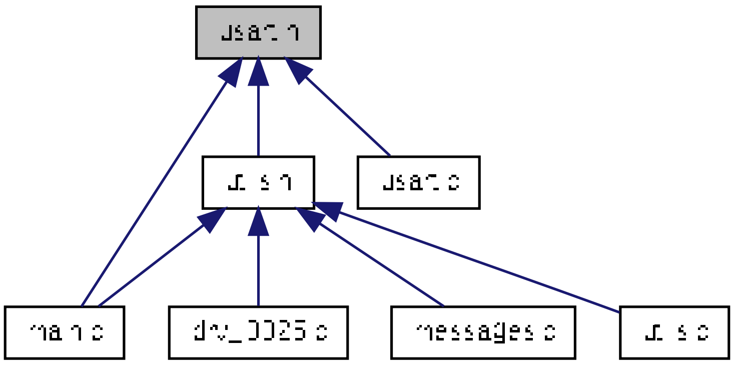
Definition at line 55 of file usart.c.

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)∗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.

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 UBRR0L 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 UCSR0A 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.

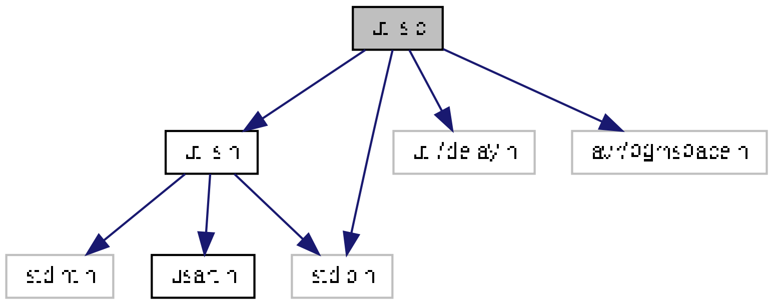
7.12.1.36 #define USBS0 USBS

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]

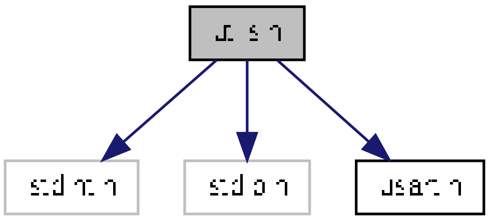
Temporarry buffer for converting uint64_t to string

Definition at line 37 of file utils.c.

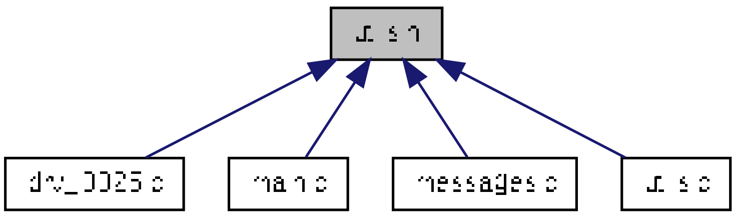
7.14 utils.h File Reference

```
#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 ConvertASCItouint64 (char ∗in)
- char ∗ int_to_string (uint64_t i)
- void putstring (const char ∗putc)
- void putstring_P (const char ∗putc)