ARM RT DSP Library

Andrew Meares

1 ARM RT DSP Library Documentation

1.1 Introduction

ARM RT DSP is a library designed to provide developers with a comprehensive set of digital signal processing (DSP) functions optimized for ARM processors. The library offers a broad range of functionalities that include but not limited to ADC operations, limit checks, delta checks, min/max functions, and various other mathematical operations common in DSP.

1.2 Examples

2 Module Documentation

2.1 PI and PID Controllers

Data Structures

- struct iir_pi_instance_q15

  Instance structure for the iir PI controller that uses a q15_t data type.
- struct iir_pi_instance_q31

  Instance structure for the iir PI controller that uses a q31_t data type.
- struct iir_pid_instance_q31

  Instance structure for the iir PID controller that uses a q31_t data type.
- struct iir_pi_instance_v2_q31

  Instance structure for the iir PI controller that uses a q31_t data type.
- struct iir_pid_instance_v2_q31

  Instance structure for the iir PI controller that uses a q31_t data type.

Macros

- #define PID_SH 4

  Defines how much the data is shifted for the iir PID controller.
- #define PID_FILTER_N 3

  Defines how much the derivative is filtered for the PID controller.

Functions

- void iir_pi_init_q15 (iir_pi_instance_q15 *S, int32_t resetStateFlag)

  Initializes PI instance structure.
- static q15_t iir_pi_q15 (iir_pi_instance_q15 *S, q15_t in)

  PI process function that uses q15_t data types.
- void iir_pi_init_q31 (iir_pi_instance_q31 *S, int32_t resetStateFlag)

  Initializes PI instance structure.
- void iir_pid_init_q31 (iir_pid_instance_q31 *S, int32_t resetStateFlag)

  Initializes PID instance structure.
- static q31_t iir_pi_q31 (iir_pi_instance_q31 *S, q31_t in)

  PI process function that uses q31_t data types.
- static q31_t iir_pid_q31 (iir_pid_instance_q31 *S, q31_t in)

  PID process function that uses q31_t data types.
- void iir_pi_init_v2_q31 (iir_pi_instance_v2_q31 *S, int32_t resetStateFlag)

  Initializes PI instance structure.
- static q31_t iir_pi_v2_q31 (iir_pi_instance_v2_q31 *S, q31_t in, int32_t b_select)

  PI process function that uses q31_t data types.
- void iir_pid_init_v2_q31 (iir_pid_instance_v2_q31 *S, int32_t resetStateFlag)

  Initializes PI instance structure.
- static q31_t iir_pid_v2_q31 (iir_pid_instance_v2_q31 *S, q31_t in, int32_t b_select)

  PI process function that uses q31_t data types.

2.1.1 Detailed Description

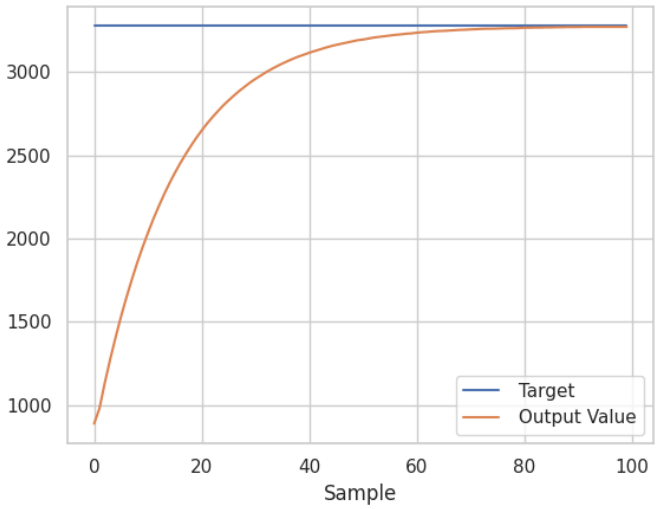


Figure 1 Simulated 1st Order Response to iir_pi_q15(...)

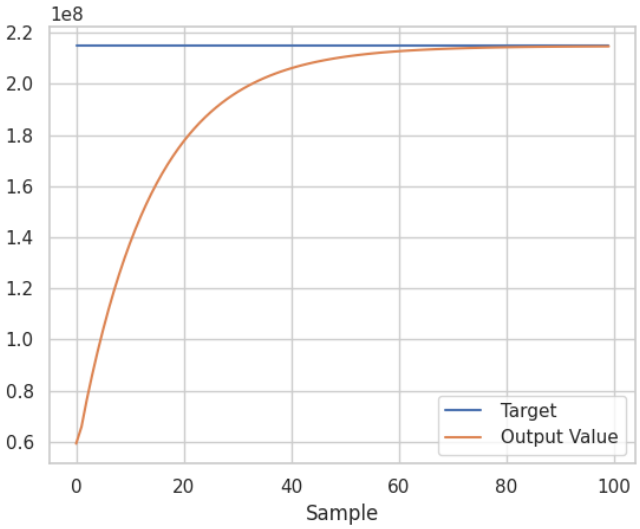


Figure 2 Simulated 1st Order Response to iir_pi_q31(...)

2.1.2 Function Documentation

2.1.2.1 iir_pi_init_q15() `void iir_pi_init_q15 (`

```
        iir_pi_instance_q15 * S,
        int32_t resetStateFlag )
```

Parameters

S	Pointer to the PI instance structure.
resetStateFlag	Set this to true to clear the state buffer.

2.1.2.2 iir_pi_init_q31() `void iir_pi_init_q31 (`

```
        iir_pi_instance_q31 * S,
        int32_t resetStateFlag )
```

Parameters

S	Pointer to the PI instance structure.
resetStateFlag	Set this to true to clear the state buffer.

2.1.2.3 iir_pi_init_v2_q31() `void iir_pi_init_v2_q31 (`

```
iir_pi_instance_v2_q31 * S,
int32_t resetStateFlag )
```

Parameters

S	Pointer to the PI instance structure.
resetStateFlag	Set this to true to clear the state buffer.

2.1.2.4 iir_pi_q15() `static q15_t iir_pi_q15 (`

```
iir_pi_instance_q15 * S,
q15_t in )  [inline], [static]
```

This is an IIR implementation of a PI controller and has practical limits on the accumulator and output. Note that if the output or accumulator saturate, the PI controller will behave poorly. A 32 bit accumulator is used to maintain as much range as possible. Gains are given in acc16_t format.

Parameters

S	Pointer to the PI instance structure.
in	Input sample value.

Returns

The controller output value.

2.1.2.5 iir_pi_q31() `static q31_t iir_pi_q31 (`

```
iir_pi_instance_q31 * S,
q31_t in )  [inline], [static]
```

This is an IIR implementation of a PI controller and has practical limits on the accumulator and output. Note that if the output or accumulator saturate, the PI controller will behave nonlinearly. A 64 bit accumulator is used to maintain as much range as possible. Gains are given in acc32_t format.

Parameters

S	Pointer to the PI instance structure.
in	Input sample value.

Returns

The controller output value.

2.1.2.6 iir_pi_v2_q31() `static q31_t iir_pi_v2_q31 (`

```
iir_pi_instance_v2_q31 * S,
q31_t in,
int32_t b_select )  [inline], [static]
```

This is an IIR implementation of a PI controller and has practical limits on the accumulator and output. Note that if the output or accumulator saturate, the PI controller will behave nonlinearly. A 64 bit accumulator is used to maintain as much range as possible. Gains are given in acc32_t format.

Parameters

S	Pointer to the PI instance structure.
in	Input sample value.
b_select	True if using alternative set of gains (B).

Returns

The controller output value.

2.1.2.7 iir_pid_init_q31() `void iir_pid_init_q31 (`

```
iir_pid_instance_q31 * S,
int32_t resetStateFlag )
```

Parameters

S	Pointer to the PI instance structure.
resetStateFlag	Set this to true to clear the state buffer.

2.1.2.8 iir_pid_init_v2_q31() `void iir_pid_init_v2_q31 (`

```
iir_pid_instance_v2_q31 * S,
int32_t resetStateFlag )
```

Parameters

S	Pointer to the PI instance structure.
resetStateFlag	Set this to true to clear the state buffer.

2.1.2.9 iir_pid_q31() `static q31_t iir_pid_q31 ( iir_pid_instance_q31 * S, q31_t in ) [inline], [static]`

This is an IIR implementation of a PID controller and has practical limits on the accumulator and output. Note that if the output or accumulator saturate, the PI controller will behave nonlinearly. A 64 bit accumulator is used to maintain as much range as possible. Gains are given in acc32_t format.

Parameters

S	Pointer to the PID instance structure.
in	Input sample value.

Returns

The controller output value.

2.1.2.10 iir_pid_v2_q31() `static q31_t iir_pid_v2_q31 ( iir_pid_instance_v2_q31 * S, q31_t in, int32_t b_select ) [inline], [static]`

This is an IIR implementation of a PI controller and has practical limits on the accumulator and output. Note that if the output or accumulator saturate, the PI controller will behave nonlinearly. A 64 bit accumulator is used to maintain as much range as possible. Gains are given in acc32_t format.

Parameters

S	Pointer to the PI instance structure.
in	Input sample value.
b_select	True if using alternative set of gains (B).

Returns

The controller output value.

2.2 Limit Functions

Functions

- static float limit_f32 (float val, float llim, float ulim)

  Limits the input value to both upper and lower limits.
- static q31_t upper_limit_q31 (q31_t val, q31_t ulim)

  Limits the input value to the supplied upper limit.

- static q31_t lower_limit_q31 (q31_t val, q31_t llim)

  Limits the input value to the supplied lower limit.
- static q31_t limit_q31 (q31_t val, q31_t llim, q31_t ulim)

  Limits the input value to the supplied upper and lower limits.
- static int32_t limit_i32 (int32_t val, int32_t llim, int32_t ulim)

  Limits the input value to the supplied upper and lower limits.
- static uint32_t limit_u32 (uint32_t val, uint32_t llim, uint32_t ulim)

  Limits the input value to the supplied upper and lower limits.
- static int16_t limit_i16 (int16_t val, int16_t llim, int16_t ulim)

  Limits the input value to the supplied upper and lower limits.
- static uint16_t limit_u16 (uint16_t val, uint16_t llim, uint16_t ulim)

  Limits the input value to the supplied upper and lower limits.
- static acc32_t limit_acc31 (acc32_t val, acc32_t llim, acc32_t ulim)

  Limits the input value to the supplied upper and lower limits.
- static uint16_t upper_limit_u16 (uint16_t val, uint16_t ulim)

  Limits the input value to the supplied upper limit.
- static q31_t max_q31 (q31_t x, q31_t y)

  Maximum function for Q31 format.
- static q31_t min_q31 (q31_t x, q31_t y)

  Minimum function for Q31 format.

2.2.1 Detailed Description

The limit functions can be used to limit the range of an input or output value in your control algorithm. The functions are inlined to avoid function call overhead. Variants allow limiting on only one side of the range or both sides.

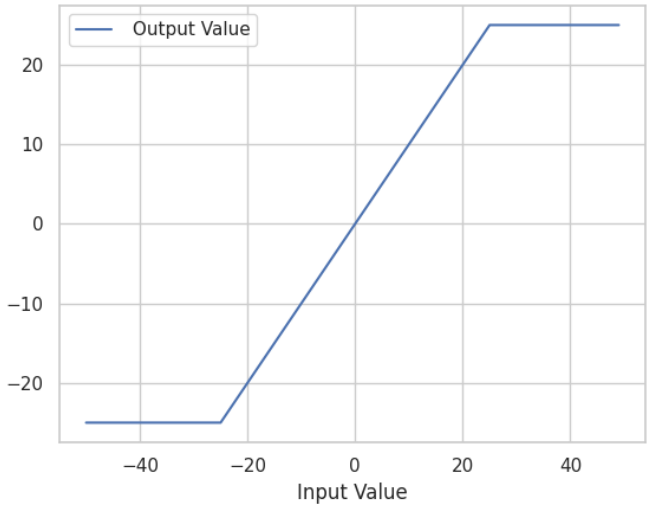


Figure 3 Sequential calls to limit_i16(val, llim, ulim)

2.2.2 Function Documentation

2.2.2.1 limit_acc31() `static acc32_t limit_acc31 ( acc32_t val, acc32_t llim, acc32_t ulim ) [inline], [static]`

Parameters

val	Input value to be limited.
llim	Lower limit to be applied.
ulim	Upper limit to be applied.

Returns

A value in the range [llim, ulim].

2.2.2.2 limit_f32() `static float limit_f32 ( float val, float llim, float ulim ) [inline], [static]`

Parameters

val	Input value to be limited.
llim	Lower limit to be applied.
ulim	Upper limit to be applied.

Returns

A value in the range [-1.0, 1.0].

2.2.2.3 limit_i16() `static int16_t limit_i16 ( int16_t val, int16_t llim, int16_t ulim ) [inline], [static]`

Parameters

val	Input value to be limited.
llim	Lower limit to be applied.
ulim	Upper limit to be applied.

Returns

A value in the range [llim, ulim].

2.2.2.4 limit_i32()

```
static int32_t limit_i32 (
        int32_t val,
        int32_t llim,
        int32_t ulim )  [inline], [static]
```

Parameters

val	Input value to be limited.
llim	Lower limit to be applied.
ulim	Upper limit to be applied.

Returns

A value in the range [llim, ulim].

2.2.2.5 limit_q31()

```
static q31_t limit_q31 (
        q31_t val,
        q31_t llim,
        q31_t ulim )  [inline], [static]
```

Parameters

val	Input value to be limited.
llim	Lower limit to be applied.
ulim	Upper limit to be applied.

Returns

A value in the range [llim, ulim].

2.2.2.6 limit_u16()

```
static uint16_t limit_u16 (
        uint16_t val,
        uint16_t llim,
        uint16_t ulim )  [inline], [static]
```

Parameters

val	Input value to be limited.
llim	Lower limit to be applied.
ulim	Upper limit to be applied.

Returns

A value in the range [llim, ulim].

2.2.2.7 limit_u32()

```
static uint32_t limit_u32 (
        uint32_t val,
        uint32_t llim,
        uint32_t ulim )  [inline], [static]
```

Parameters

val	Input value to be limited.
llim	Lower limit to be applied.
ulim	Upper limit to be applied.

Returns

A value in the range [llim, ulim].

2.2.2.8 lower_limit_q31()

```
static q31_t lower_limit_q31 (
        q31_t val,
        q31_t llim )  [inline], [static]
```

Parameters

val	Input value to be limited.
llim	Lower limit to be applied.

Returns

A value in the range [llim, 1.0].

2.2.2.9 max_q31() static q31_t max_q31 (q31_t *x*, q31_t *y*) [inline], [static]

Parameters

x	First value.
y	Second value.

Returns

The maximum of x and y.

2.2.2.10 min_q31() static q31_t min_q31 (q31_t *x*, q31_t *y*) [inline], [static]

Parameters

x	First value.
y	Second value.

Returns

The minimum of x and y.

2.2.2.11 upper_limit_q31() static q31_t upper_limit_q31 (q31_t *val*, q31_t *ulim*) [inline], [static]

Parameters

val	Input value to be limited.
ulim	Upper limit to be applied.

Returns

A value in the range [-1.0, ulim].

2.2.2.12 upper_limit_u16() `static uint16_t upper_limit_u16 ( uint16_t val, uint16_t ulim ) [inline], [static]`

Parameters

val	Input value to be limited.
ulim	Upper limit to be applied.

Returns

A value in the range [0, ulim].

3 Data Structure Documentation

3.1 filter_pma_a63_t Struct Reference

Pseudo windowed moving average data structure.

`#include <arm_rt_dsp_filter.h>`

Data Fields

- acc64_t acc
  A 64-bit accumulator.
- uint16_t sh
  The window size is equal to 2^ sh.

3.1.1 Detailed Description

The init function from the 56800EX DSP library sets the accumulator to a value that would output the desired initial value on the first iteration. For now, just initialize the the structure to the desired window size and zero the accumulator. Remember the accumulator will have to wind up when the filter is used. For glacially slow filters, winding up might take eons.

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

- arm_rt_dsp_filter.h

3.2 hysteresis_thresh_i16_t Struct Reference

Context structure for the hysteresis function.

`#include <arm_rt_dsp_misc.h>`

Data Fields

- int16_t hyst_on
- int16_t hyst_off
- int16_t out_state

3.2.1 Field Documentation

3.2.1.1 hyst_off `int16_t hyst_off`

Value determining the lower threshold

3.2.1.2 hyst_on `int16_t hyst_on`

Value determining the upper threshold

3.2.1.3 out_state `int16_t out_state`

Actual state of the output

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

- arm_rt_dsp_misc.h

3.3 hysteresis_thresh_t Struct Reference

Context structure for the hysteresis function.

```
#include <arm_rt_dsp_misc.h>
```

Data Fields

- q31_t hyst_on
- q31_t hyst_off
- int32_t out_state

3.3.1 Field Documentation

3.3.1.1 hyst_off `q31_t hyst_off`

Value determining the lower threshold

3.3.1.2 hyst_on `q31_t hyst_on`

Value determining the upper threshold

3.3.1.3 out_state `int32_t out_state`

Actual state of the output

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

- arm_rt_dsp_misc.h

3.4 iir_pi_instance_q15 Struct Reference

Instance structure for the iir PI controller that uses a q15_t data type.

`#include <arm_rt_dsp_controller.h>`

3.4.1 Detailed Description

To initialize, set the gains Kp and Ki. The derived gains are calculated as A0 = Kp + Ki and A1 = -Kp and the state buffer is cleared in the init function.

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

- arm_rt_dsp_controller.h

3.5 iir_pi_instance_q31 Struct Reference

Instance structure for the iir PI controller that uses a q31_t data type.

`#include <arm_rt_dsp_controller.h>`

3.5.1 Detailed Description

To initialize, set the gains Kp and Ki. The derived gains are calculated as A0 = Kp + Ki and A1 = -Kp and the state buffer is cleared in the init function.

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

- arm_rt_dsp_controller.h

3.6 iir_pi_instance_v2_q31 Struct Reference

Instance structure for the iir PI controller that uses a q31_t data type.

`#include <arm_rt_dsp_controller.h>`

3.6.1 Detailed Description

To initialize, set the gains Kp, Ki, and Kd. The derived gains are calculated as A0 = Kp + Ki + Kd and A1 = -Kp - 2Kd and the state buffer is cleared in the init function.

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

- arm_rt_dsp_controller.h

3.7 iir_pid_instance_q31 Struct Reference

Instance structure for the iir PID controller that uses a q31_t data type.

`#include <arm_rt_dsp_controller.h>`

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

- arm_rt_dsp_controller.h

3.8 iir_pid_instance_v2_q31 Struct Reference

Instance structure for the iir PI controller that uses a q31_t data type.

`#include <arm_rt_dsp_controller.h>`

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

- arm_rt_dsp_controller.h

3.9 ramp_limit_i16_t Struct Reference

Signed int16_t ramp limiter data structure.

`#include <arm_rt_dsp_ramp.h>`

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

- arm_rt_dsp_ramp.h

3.10 ramp_limit_q15_t Struct Reference

Signed q15_t ramp limiter data structure.

`#include <arm_rt_dsp_ramp.h>`

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

- arm_rt_dsp_ramp.h

3.11 ramp_q31_t Struct Reference

Signed q31_t ramp data structure.

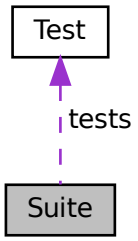
`#include <arm_rt_dsp_ramp.h>`

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

- arm_rt_dsp_ramp.h

3.12 Suite Struct Reference

Collaboration diagram for Suite:



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

- common.h

3.13 Test Struct Reference

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

- common.h

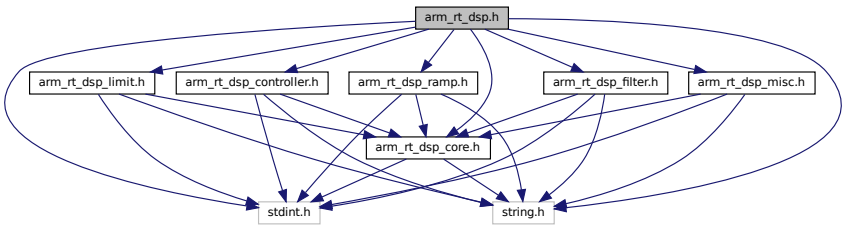
4 File Documentation

4.1 arm_rt_dsp.h File Reference

The main header file for the ARM RT DSP library.

```
#include <stdint.h>
#include <string.h>
#include "arm_rt_dsp_core.h"
#include "arm_rt_dsp_limit.h"
#include "arm_rt_dsp_filter.h"
#include "arm_rt_dsp_ramp.h"
#include "arm_rt_dsp_controller.h"
#include "arm_rt_dsp_misc.h"
```

Include dependency graph for arm_rt_dsp.h:



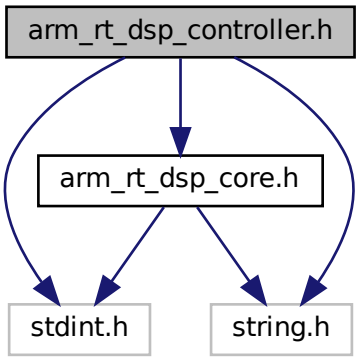
4.2 arm_rt_dsp_controller.h File Reference

Controllers.

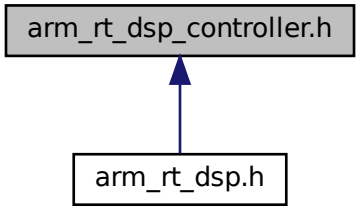
```
#include <stdint.h>
#include <string.h>
```

```
#include "arm_rt_dsp_core.h"
```

Include dependency graph for arm_rt_dsp_controller.h:



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



Data Structures

- struct iir_pi_instance_q15

  Instance structure for the iir PI controller that uses a q15_t data type.
- struct iir_pi_instance_q31

  Instance structure for the iir PI controller that uses a q31_t data type.
- struct iir_pid_instance_q31

  Instance structure for the iir PID controller that uses a q31_t data type.
- struct iir_pi_instance_v2_q31

  Instance structure for the iir PI controller that uses a q31_t data type.
- struct iir_pid_instance_v2_q31

  Instance structure for the iir PI controller that uses a q31_t data type.

Macros

- #define PID_SH 4

  Defines how much the data is shifted for the iir PID controller.
- #define PID_FILTER_N 3

  Defines how much the derivative is filtered for the PID controller.

Functions

- void iir_pi_init_q15 (iir_pi_instance_q15 *S, int32_t resetStateFlag)

  Initializes PI instance structure.
- static q15_t iir_pi_q15 (iir_pi_instance_q15 *S, q15_t in)

  PI process function that uses q15_t data types.
- void iir_pi_init_q31 (iir_pi_instance_q31 *S, int32_t resetStateFlag)

  Initializes PI instance structure.
- void iir_pid_init_q31 (iir_pid_instance_q31 *S, int32_t resetStateFlag)

  Initializes PID instance structure.
- static q31_t iir_pi_q31 (iir_pi_instance_q31 *S, q31_t in)

  PI process function that uses q31_t data types.
- static q31_t iir_pid_q31 (iir_pid_instance_q31 *S, q31_t in)

  PID process function that uses q31_t data types.
- void iir_pi_init_v2_q31 (iir_pi_instance_v2_q31 *S, int32_t resetStateFlag)

  Initializes PI instance structure.
- static q31_t iir_pi_v2_q31 (iir_pi_instance_v2_q31 *S, q31_t in, int32_t b_select)

  PI process function that uses q31_t data types.
- void iir_pid_init_v2_q31 (iir_pid_instance_v2_q31 *S, int32_t resetStateFlag)

  Initializes PI instance structure.
- static q31_t iir_pid_v2_q31 (iir_pid_instance_v2_q31 *S, q31_t in, int32_t b_select)

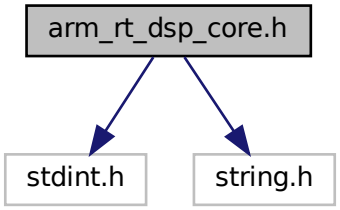
  PI process function that uses q31_t data types.

4.3 arm_rt_dsp_core.h File Reference

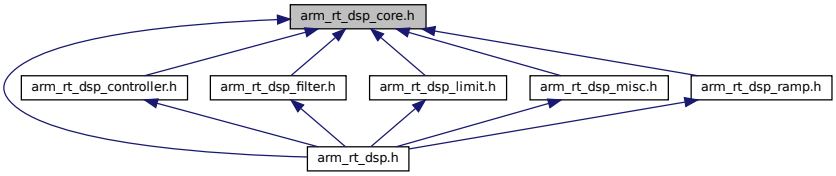
This file contains the definitions of the ARM DSP types and functions.

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

Include dependency graph for arm_rt_dsp_core.h:



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



Macros

- #define Q15(x) ((q15_t)((x) < 0.999969482421875 ? ((x) >= -1 ? (x)*0x8000 : 0x8000) : 0x7FFF))

  Macro for defining a q15_t constant value in the range [-1.0, 1.0).
- #define Q31(x) ((q31_t)((x) < 1 ? ((x) >= -1 ? (x)*0x80000000 : 0x80000000) : 0x7FFFFFFF))

  Macro for defining a q31_t constant value in the range [-1.0, 1.0).
- #define ACC16(x) ((acc16_t)((x) < 255.9921875 ? ((x) >= -256 ? (x)*0x80 : 0x8000) : 0x7FFF))

  Macro for defining an acc16_t constant value in the range [-255.0, 255.0).
- #define ACC32(x) ((acc32_t)((x) < 65535.999969482421875 ? ((x) >= -65536 ? (x)*0x8000 : 0x80000000) : 0x7FFFFFFF))

  Macro for defining an acc32_t constant value in the range [-65535.0, 65535.0).

Typedefs

- typedef int64_t q63_t

  A 64-bit fractional data type in 1.63 format.
- typedef int16_t acc16_t

  A 16-bit accumulator data type in 9.7 format.
- typedef int32_t acc32_t

  A 32-bit accumulator data type in 17.15 format.
- typedef int64_t acc64_t

  A 64-bit accumulator in 33.31 format.

Functions

- static int64_t ssat_i64 (int64_t val, uint32_t sat)

  Signed Saturate.
- static q15_t abs_q15 (q15_t parVal)

  Calculates the absolute value of the input without saturation.
- static q31_t abs_q31 (q31_t parVal)

  Calculates the absolute value of the input without saturation.
- static q15_t abs_sat_q15 (q15_t parVal)

  Calculates the absolute value of the input without saturation.
- static q31_t abs_sat_q31 (q31_t parVal)

Calculates the absolute value of the input with saturation.

- static q15_t mul_q15 (q15_t x, q15_t y)

  Multiplies two Q15s.
- static q31_t mul_q31 (q31_t x, q31_t y)

  Multiplies two Q31s.
- static q15_t mulsat_q15 (q15_t x, q15_t y)

  Multiplies two Q15s with saturation.
- static q31_t mulsat_q31 (q31_t x, q31_t y)

  Multiplies two Q31s with saturation.

4.3.1 Typedef Documentation

4.3.1.1 acc16_t `typedef int16_t acc16_t`

In order to stay consistent with the ARM DSP Q types, the sign bit is not counted.

4.3.1.2 acc32_t `typedef int32_t acc32_t`

In order to stay consistent with the ARM DSP Q types, the sign bit is not counted.

Does this name even make sense. Maybe acc32_t was better or acc17_15_t or anything. AM

4.3.1.3 acc64_t `typedef int64_t acc64_t`

In order to stay consistent with the ARM DSP Q types, the sign bit is not counted.

4.3.1.4 q63_t `typedef int64_t q63_t`

The most significant bit is the sign bit.

4.3.2 Function Documentation

4.3.2.1 abs_q15() `static q15_t abs_q15 ( q15_t parVal ) [inline], [static]`

Parameters

parVal	The input value.

Returns

The absolute value of the input in the range [1.0, 1.0).

4.3.2.2 abs_q31() `static q31_t abs_q31 ( q31_t parVal ) [inline], [static]`

Parameters

parVal	The input value.

Returns

The absolute value of the input in the range [1.0, 1.0).

4.3.2.3 abs_sat_q15() `static q15_t abs_sat_q15 ( q15_t parVal ) [inline], [static]`

Parameters

parVal	The input value.

Returns

The absolute value of the input in the range [1.0, 1.0).

4.3.2.4 abs_sat_q31() `static q31_t abs_sat_q31 ( q31_t parVal ) [inline], [static]`

Parameters

parVal	The input value.

Returns

The absolute value of the input saturated to [1.0, 1.0).

4.3.2.5 mul_q15() `static q15_t mul_q15 ( q15_t x, q15_t y ) [inline], [static]`

Parameters

x	The first multiplicand.
y	The second multiplicand.

Returns

The product of the two multiplicands.

4.3.2.6 mul_q31() `static q31_t mul_q31 ( q31_t x, q31_t y ) [inline], [static]`

Parameters

x	The first multiplicand.
y	The second multiplicand.

Returns

The product of the two multiplicands.

4.3.2.7 mulsat_q15() `static q15_t mulsat_q15 ( q15_t x, q15_t y ) [inline], [static]`

Some of these capabilities are captured in the toolchain's intrinsics or "idioms".

Parameters

x	The first multiplicand.
y	The second multiplicand.

Returns

The saturated product of the two multiplicands with a range of [-1.0, 1.0).

4.3.2.8 mulsat_q31() `static q31_t mulsat_q31 ( q31_t x, q31_t y ) [inline], [static]`

Parameters

x	The first multiplicand.
y	The second multiplicand.

Returns

The saturated product of the two multiplicands with a range of [-1.0, 1.0).

4.3.2.9 ssat_i64() `static int64_t ssat_i64 ( int64_t val, uint32_t sat ) [inline], [static]`

Saturates a signed value.

Parameters

in	*value*	Value to be saturated
in	*sat*	Bit position to saturate to (1..32)

Returns

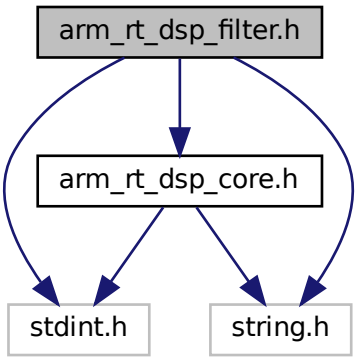
Saturated value

4.4 arm_rt_dsp_filter.h File Reference

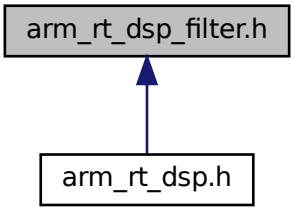
This file contains the definitions of the ARM DSP types and functions.

```
#include <stdint.h>
#include <string.h>
#include "arm_rt_dsp_core.h"
```

Include dependency graph for arm_rt_dsp_filter.h:



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



Data Structures

- struct filter_pma_a63_t

  Pseudo windowed moving average data structure.

Functions

- static q31_t filter_pma_q31 (q31_t inx, filter_pma_a63_t *param)

  A process function for a pseudo windowed moving average filter.

4.4.1 Function Documentation

4.4.1.1 filter_pma_q31() `static q31_t filter_pma_q31 ( q31_t inx, filter_pma_a63_t * param ) [inline], [static]`

Parameters

inx	The new input sample.
param	The filter's configuration and state data.

Returns

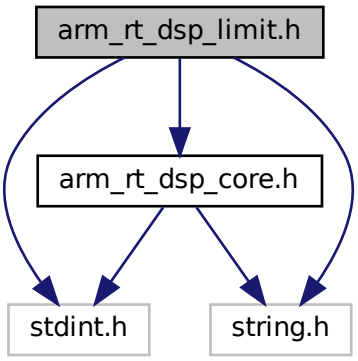
A new filtered output sample.

4.5 arm_rt_dsp_limit.h File Reference

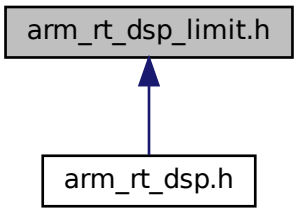
This file contains the definitions of the ARM DSP types and functions.

```
#include <stdint.h>
#include <string.h>
#include "arm_rt_dsp_core.h"
```

Include dependency graph for arm_rt_dsp_limit.h:



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



Functions

- static float limit_f32 (float val, float llim, float ulim)

  Limits the input value to both upper and lower limits.
- static q31_t upper_limit_q31 (q31_t val, q31_t ulim)

  Limits the input value to the supplied upper limit.
- static q31_t lower_limit_q31 (q31_t val, q31_t llim)

  Limits the input value to the supplied lower limit.
- static q31_t limit_q31 (q31_t val, q31_t llim, q31_t ulim)

  Limits the input value to the supplied upper and lower limits.
- static int32_t limit_i32 (int32_t val, int32_t llim, int32_t ulim)

  Limits the input value to the supplied upper and lower limits.
- static uint32_t limit_u32 (uint32_t val, uint32_t llim, uint32_t ulim)

  Limits the input value to the supplied upper and lower limits.
- static int16_t limit_i16 (int16_t val, int16_t llim, int16_t ulim)

  Limits the input value to the supplied upper and lower limits.
- static uint16_t limit_u16 (uint16_t val, uint16_t llim, uint16_t ulim)

  Limits the input value to the supplied upper and lower limits.
- static acc32_t limit_acc31 (acc32_t val, acc32_t llim, acc32_t ulim)

  Limits the input value to the supplied upper and lower limits.
- static uint16_t upper_limit_u16 (uint16_t val, uint16_t ulim)

  Limits the input value to the supplied upper limit.
- static q31_t max_q31 (q31_t x, q31_t y)

  Maximum function for Q31 format.
- static q31_t min_q31 (q31_t x, q31_t y)

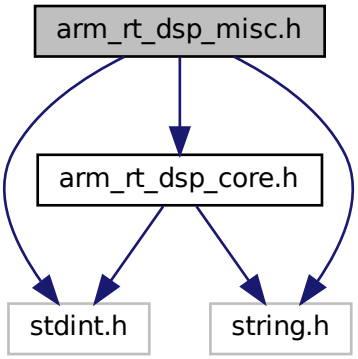
  Minimum function for Q31 format.

4.6 arm_rt_dsp_misc.h File Reference

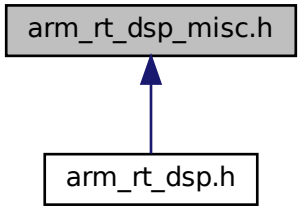
This file contains the definitions of the ARM DSP types and functions.

```
#include <stdint.h>
#include <string.h>
#include "arm_rt_dsp_core.h"
```

Include dependency graph for arm_rt_dsp_misc.h:



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



Data Structures

- struct hysteresis_thresh_t

  Context structure for the hysteresis function.
- struct hysteresis_thresh_i16_t

  Context structure for the hysteresis function.

Functions

- static q15_t adc_process_sample_q15 (uint16_t x, int16_t offset, q15_t slope)

  Converts an unsigned ADC raw value to a Q15 format with offset and scale applied.
- static q31_t adc_process_sample_q31 (uint16_t x, int16_t offset, q31_t slope)

  Converts an unsigned ADC raw value to a Q31 format with offset and scale applied.
- static q31_t adc_process_sample_u_q31 (uint16_t x, int16_t offset, q31_t slope)

  Converts an unsigned ADC raw value to a Q31 format with offset and scale applied.
- static q31_t adc_process_sample_i16_q31 (int16_t x, int16_t offset, q31_t slope)

  Converts an unsigned ADC raw value to a Q31 format with offset and scale applied.
- static int16_t convert_round_q31_to_i16 (q31_t x, uint32_t scale)

  Converts a value from 1.31 fixed point format, q31_t, to an int16_t with an applied scaling factor and rounding.
- static int16_t convert_q31_to_i16 (q31_t x, uint32_t scale)

  Converts a value from 1.31 fixed point format, q31_t, to an int16_t with an applied scaling factor.
- static uint16_t convert_q31_to_u16 (q31_t x, uint32_t scale)

  Converts a value from 1.31 fixed point format, q31_t, to a uint16_t with an applied scaling factor.
- static uint32_t convert_q31_to_u32 (q31_t x, uint32_t scale)

  Converts a value from 1.31 fixed point format, q31_t, to a uint32_t with an applied scaling factor.
- static int32_t convert_round_q31_to_i32 (q31_t x, uint32_t scale)

  Converts a value from 1.31 fixed point format, q31_t, to a int32_t with an applied scaling factor and rounding.
- static int32_t convert_q31_to_i32 (q31_t x, uint32_t scale)

  Converts a value from 1.31 fixed point format, q31_t, to a int32_t with an applied scaling factor.
- static q31_t convert_u16_to_q31 (uint16_t x, int32_t scale)

  Convert a uint16_t to a q31_t.
- static q31_t convert_i16_to_q31 (int16_t x, int32_t scale)

  Convert an int16_t to a q31_t.
- static q31_t convert_i32_to_q31 (int32_t x, int32_t scale)

  Convert an int32_t to a q31_t.
- void hysteresis_init (q31_t l_thresh, q31_t h_thresh, hysteresis_thresh_t *H)

  Initialize a hyseresis process function.
- static int32_t hysteresis_threshold (q31_t val, hysteresis_thresh_t *H)

  Applies a threshold with hysteresis to a q31_t value.
- void hysteresis_init_i16 (int16_t l_thresh, int16_t h_thresh, hysteresis_thresh_i16_t *H)

  Initialize a hyseresis process function.
- static int32_t hysteresis_threshold_i16 (int16_t val, hysteresis_thresh_i16_t *H)

  Applies a threshold with hysteresis to a i16 value.
- static int32_t check_delta_q31 (q31_t value, q31_t nominal, q31_t delta)

  Checks if a value is within some delta of a nominal value.
- static int32_t check_delta_f32 (float32_t value, float32_t nominal, float32_t delta)

  Checks if a value is within some delta of a nominal value.

4.6.1 Function Documentation

4.6.1.1 adc_process_sample_i16_q31()

```
static q31_t adc_process_sample_i16_q31 (
        int16_t x,
        int16_t offset,
        q31_t slope )  [inline], [static]
```

This version is for inputs that use 0V as the reference.

Parameters

x	The raw ADC result value in "counts".
offset	The offset is specified as an unsigned "counts" value.
slope	The slope is specified in Q31 format and is in the range [-1.0, 1.0).

Returns

The offset and scaled result in Q31 format is in the range [-1.0, 1.0).

4.6.1.2 adc_process_sample_q15()

```
static q15_t adc_process_sample_q15 (
        uint16_t x,
        int16_t offset,
        q15_t slope )  [inline], [static]
```

The raw value must be provided in right justified format and is assumed to be 12 bits. Saturation arithmetic is used.

Parameters

x	The raw ADC result value in "counts".
offset	The offset is specified as an unsigned "counts" value.
slope	The slope is specified in Q15 format and is in the range [-1.0, 1.0).

Returns

The offset and scaled result in Q15 format is in the range [-1.0, 1.0).

4.6.1.3 adc_process_sample_q31()

```
static q31_t adc_process_sample_q31 (
        uint16_t x,
        int16_t offset,
        q31_t slope )  [inline], [static]
```

This version is for input that use the mid-rail as the reference. The raw value must be provided in right justified format and is assumed to be 12 bits. Saturation arithmetic is used.

Parameters

x	The raw ADC result value in "counts".
offset	The offset is specified as an unsigned "counts" value.
slope	The slope is specified in Q31 format and is in the range [-1.0, 1.0).

Returns

The offset and scaled result in Q31 format is in the range [-1.0, 1.0).

4.6.1.4 adc_process_sample_u_q31()

```
static q31_t adc_process_sample_u_q31 (
        uint16_t x,
        int16_t offset,
        q31_t slope )  [inline], [static]
```

This version is for inputs that use 0V as the reference.

Parameters

x	The raw ADC result value in "counts".
offset	The offset is specified as an unsigned "counts" value.
slope	The slope is specified in Q31 format and is in the range [-1.0, 1.0).

Returns

The offset and scaled result in Q31 format is in the range [-1.0, 1.0).

4.6.1.5 check_delta_f32()

```
static int32_t check_delta_f32 (
        float32_t value,
        float32_t nominal,
        float32_t delta )  [inline], [static]
```

Parameters

value	The input value.
nominal	The nominal value.
delta	The delta applied above and below the nominal value.

Returns

True if nominal - delta $<$ value $<$ nominal + delta. Otherwise returns false.

4.6.1.6 check_delta_q31()

```
static int32_t check_delta_q31 (
        q31_t value,
        q31_t nominal,
        q31_t delta )  [inline], [static]
```

Parameters

value	The input value.
nominal	The nominal value.
delta	The delta applied above and below the nominal value.

Returns

True if nominal - delta $<$ value $<$ nominal + delta. Otherwise returns false.

4.6.1.7 convert_i16_to_q31()

```
static q31_t convert_i16_to_q31 (
        int16_t x,
        int32_t scale )  [inline], [static]
```

Parameters

x	Value to convert.
mul	Scale of value to convert.

Returns

Converted value.

4.6.1.8 convert_i32_to_q31()

```
static q31_t convert_i32_to_q31 (
        int32_t x,
        int32_t scale )  [inline], [static]
```

Parameters

x	Value to convert.
mul	Scale of value to convert.

Returns

Converted value.

4.6.1.9 convert_q31_to_i16()

```
static int16_t convert_q31_to_i16 (
        q31_t x,
        uint32_t scale )  [inline], [static]
```

Parameters

x	Input value to be converted
scale	Scaling value to be applied.

Returns

The converted value.

4.6.1.10 convert_q31_to_i32()

```
static int32_t convert_q31_to_i32 (
    q31_t x,
    uint32_t scale )  [inline], [static]
```

Parameters

x	Input value to be converted
scale	Scaling value to be applied.

Returns

The converted value.

4.6.1.11 convert_q31_to_u16()

```
static uint16_t convert_q31_to_u16 (
    q31_t x,
    uint32_t scale )  [inline], [static]
```

Parameters

x	Input value to be converted
scale	Scaling value to be applied.

Returns

The converted value.

4.6.1.12 convert_q31_to_u32()

```
static uint32_t convert_q31_to_u32 (
    q31_t x,
    uint32_t scale )  [inline], [static]
```

Parameters

x	Input value to be converted
scale	Scaling value to be applied.

Returns

The converted value.

4.6.1.13 convert_round_q31_to_i16()

```
static int16_t convert_round_q31_to_i16 (
    q31_t x,
    uint32_t scale ) [inline], [static]
```

A voltage measurement in q31_t format with a scale of 1.0 represents 1kV has a range of [-1.0kV, 1.0kV). Calling convert_round_q31_to_i16 on this measurement with a scale of 1000 will yield an integer in the range of [-1000, 1000) with units of volts.

Parameters

x	Input value to be converted
scale	Scaling value to be applied.

Returns

The converted value.

4.6.1.14 convert_round_q31_to_i32()

```
static int32_t convert_round_q31_to_i32 (
    q31_t x,
    uint32_t scale ) [inline], [static]
```

Parameters

x	Input value to be converted
scale	Scaling value to be applied.

Returns

The converted value.

4.6.1.15 convert_u16_to_q31() `static q31_t convert_u16_to_q31 (`

```
uint16_t x,
int32_t scale )  [inline], [static]
```

Parameters

x	Value to convert.
mul	Scale of value to convert.

Returns

Converted value.

4.6.1.16 hysteresis_init() `void hysteresis_init (`

```
q31_t l_thresh,
q31_t h_thresh,
hysteresis_thresh_t * H )
```

Parameters

l_thresh	Value determining the lower threshold.
h_thresh	Value determining the upper threshold.
H	The context structure for this hysteresis process.

4.6.1.17 hysteresis_init_i16() `void hysteresis_init_i16 (`

```
int16_t l_thresh,
int16_t h_thresh,
hysteresis_thresh_i16_t * H )
```

Parameters

l_thresh	Value determining the lower threshold.
h_thresh	Value determining the upper threshold.
H	The context structure for this hysteresis process.

4.6.1.18 hysteresis_threshold() `static int32_t hysteresis_threshold (`

```
q31_t val,
hysteresis_thresh_t * H )  [inline], [static]
```

Parameters

val	The input value.
H	The context structure for this hysteresis process.

Returns

True or false.

4.6.1.19 hysteresis_threshold_i16()

```
static int32_t hysteresis_threshold_i16 (
        int16_t val,
        hysteresis_thresh_i16_t * H )  [inline], [static]
```

Parameters

val	The input value.
H	The context structure for this hysteresis process.

Returns

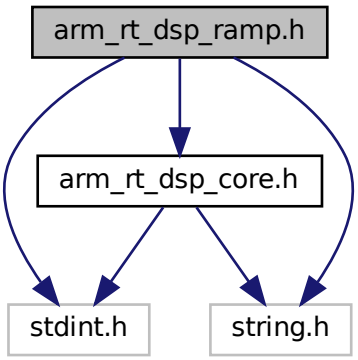
True or false.

4.7 arm_rt_dsp_ramp.h File Reference

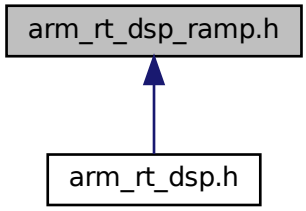
Ramp functions.

```
#include <stdint.h>
#include <string.h>
#include "arm_rt_dsp_core.h"
```

Include dependency graph for arm_rt_dsp_ramp.h:



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



Data Structures

- struct ramp_limit_i16_t

  Signed int16_t ramp limiter data structure.
- struct ramp_limit_q15_t

  Signed q15_t ramp limiter data structure.
- struct ramp_q31_t

  Signed q31_t ramp data structure.

Functions

- void ramp_limit_init_i16 (int16_t y0, ramp_limit_i16_t ∗r)

  Initialize the linear ramp data structure with an initial output value.
- int16_t ramp_limit_i16 (int16_t x, ramp_limit_i16_t ∗r)

  Linear ramp from one number to another with upper limit and lower limit applied.
- void ramp_limit_init_q15 (q15_t y0, ramp_limit_q15_t ∗r)

  Initialize the linear ramp data structure with an initial output value.
- q15_t ramp_limit_q15 (q15_t x, ramp_limit_q15_t ∗r)

  Linear ramp from one number to another with upper limit and lower limit applied.
- void ramp_init_q31 (q31_t y0, ramp_q31_t ∗r)

  Initialize the linear ramp data structure with an initial output value.
- q31_t ramp_q31 (q31_t x, ramp_q31_t ∗r)

  Linear ramp from one number to another.

4.7.1 Function Documentation

4.7.1.1 ramp_init_q31()

```
void ramp_init_q31 (
        q31_t y0,
        ramp_q31_t * r )
```

Parameters

y0	Initial output value.
r	Ramp data structure.

4.7.1.2 ramp_limit_i16()

```
int16_t ramp_limit_i16 (
        int16_t x,
        ramp_limit_i16_t * r )
```

Parameters

x	Input value is new or current ramp target.
r	Ramp data structure.

Returns

The ramped value output approaches the current target at the ramp rate specified.

4.7.1.3 ramp_limit_init_i16() `void ramp_limit_init_i16 (`

```
int16_t y0,
ramp_limit_i16_t * r )
```

Parameters

y0	Initial output value.
r	Ramp data structure.

4.7.1.4 ramp_limit_init_q15() `void ramp_limit_init_q15 (`

```
q15_t y0,
ramp_limit_q15_t * r )
```

Parameters

y0	Initial output value.
r	Ramp data structure.

4.7.1.5 ramp_limit_q15() `q15_t ramp_limit_q15 (`

```
q15_t x,
ramp_limit_q15_t * r )
```

Parameters

x	Input value is new or current ramp target.
r	Ramp data structure.

Returns

The ramped value output approaches the current target at the ramp rate specified.

4.7.1.6 ramp_q31() `q31_t ramp_q31 (`

```
q31_t x,
ramp_q31_t * r )
```

Parameters

x	Input value is new or current ramp target.
r	Ramp data structure.

Returns

The ramped value output approaches the current target at the ramp rate specified.

4.8 common.h File Reference

Common definitions and additional documentation for unit tests.

Data Structures

- struct Test
- struct Suite

Index