MindMotion
SoC Solutions

MLIB

应用笔记

Rev 1.0　2022/04/14

MLIB 用户指南

简介

该文档用于描述基于灵动微电子的 Arm® Cortex®-M0 内核的 MCU 电机控制数学库(MLIB)。

目录

1 引言 8
1.1 概述 8
1.2 数据类型 8
1.3 例程定义 8
2 库函数说明 9
2.1 MLIB_Neg_Q15 9
2.1.1 函数说明 9
2.1.2 描述 9
2.1.3 使用例程 9
2.2 MLIB_Neg_Q31 9
2.2.1 函数说明 9
2.2.2 描述 10
2.2.3 使用例程 10
2.3 MLIB_NegSat_Q31 10
2.3.1 函数说明 10
2.3.2 描述 10
2.3.3 使用例程 10
2.4 MLIB_Abs_Q15 11
2.4.1 函数说明 11
2.4.2 描述 11
2.4.3 使用例程 11
2.5 MLIB_AbsSat_Q15 11
2.5.1 函数说明 11
2.5.2 描述 11
2.5.3 使用例程 12
2.6 MLIB_Abs_Q31 12
2.6.1 函数说明 12
2.6.2 描述 12

2.6.3 使用例程 12
2.7 MLIB_AbsSat_Q31 12
2.7.1 函数说明 13
2.7.2 描述 13
2.7.3 使用例程 13
2.8 MLIB_Add_Q15 13
2.8.1 函数说明 13
2.8.2 描述 13
2.8.3 使用例程 14
2.9 MLIB_AddSat_Q15 14
2.9.1 函数说明 14
2.9.2 描述 14
2.9.3 使用例程 14
2.10 MLIB_Add_Q31 15
2.10.1 函数说明 15
2.10.2 描述 15
2.10.3 使用例程 15
2.11 MLIB_AddSat_Q31 15
2.11.1 函数说明 15
2.11.2 描述 15
2.11.3 使用例程 16
2.12 MLIB_ShL_Q15 16
2.12.1 函数说明 16
2.12.2 描述 16
2.12.3 使用例程 16
2.13 MLIB_ShR_Q15 17
2.13.1 函数说明 17
2.13.2 描述 17
2.13.3 使用例程 17

2.14 MLIB_ShLBi_Q15 17
2.14.1 函数说明 17
2.14.2 描述 18
2.14.3 使用例程 18
2.15 MLIB_ShRBi_Q15 18
2.15.1 函数说明 18
2.15.2 描述 18
2.15.3 使用例程 18
2.16 MLIB_Sub_Q15 19
2.16.1 函数说明 19
2.16.2 描述 19
2.16.3 使用例程 19
2.17 MLIB_SubSat_Q15 19
2.17.1 函数说明 20
2.17.2 描述 20
2.17.3 使用例程 20
2.18 MLIB_Sub_Q31 20
2.18.1 函数说明 20
2.18.2 描述 20
2.18.3 使用例程 21
2.19 MLIB_SubSat_Q31 21
2.19.1 函数说明 21
2.19.2 描述 21
2.19.3 使用例程 21
2.20 MLIB_ShL_Q31 22
2.20.1 函数说明 22
2.20.2 描述 22
2.20.3 使用例程 22
2.21 MLIB_ShR_Q31 22

2.21.1 函数说明 22
2.21.2 描述 23
2.21.3 使用例程 23
2.22 MLIB_ShLBi_Q31 23
2.22.1 函数说明 23
2.22.2 描述 23
2.22.3 使用例程 23
2.23 MLIB_ShRBi_Q31 24
2.23.1 函数说明 24
2.23.2 描述 24
2.23.3 使用例程 24
2.24 MLIB_ShLSat_Q31 24
2.24.1 函数说明 24
2.24.2 描述 25
2.24.3 使用例程 25
2.25 MLIB_Conv_Q31s 25
2.25.1 函数说明 25
2.25.2 描述 25
2.25.3 使用例程 25
2.26 MLIB_Conv_Q15s 26
2.26.1 函数说明 26
2.26.2 描述 26
2.26.3 使用例程 26
2.27 MLIB_Mul_Q15 26
2.27.1 函数说明 26
2.27.2 描述 27
2.27.3 使用例程 27
2.28 MLIB_Mul_Q31 27
2.28.1 函数说明 27

2.28.2 描述 27
2.28.3 使用例程 27
2.29 MLIB_Mul_Q31ss 28
2.29.1 函数说明 28
2.29.2 描述 28
2.29.3 使用例程 28
2.30 MLIB_MulSat_Q31 28
2.30.1 函数说明 28
2.30.2 描述 29
2.30.3 使用例程 29
2.31 MLIB_MulSat_Q31ss 29
2.31.1 函数说明 29
2.31.2 描述 29
2.31.3 使用例程 29
2.32 MLIB_Mul_Q31s 30
2.32.1 函数说明 30
2.32.2 描述 30
2.32.3 使用例程 30
2.33 MLIB_FilterMA_Q15 30
2.33.1 函数说明 31
2.33.2 描述 31
2.33.3 使用例程 31
2.34 MFLIB_SMO 31
2.34.1 函数说明 32
2.34.2 描述 33
2.34.3 使用例程 33
2.35 MLIB_PLL 34
2.35.1 函数说明 35
2.35.2 描述 35

2.35.3 使用例程 36
2.36 MLIB_PIControl_Q15 36
2.36.1 函数说明 36
2.36.2 描述 37
2.36.3 使用例程 37
2.37 MLIB_Ramp_Q15 37
2.37.1 函数说明 38
2.37.2 描述 38
2.37.3 使用例程 38
2.38 MLIB_Sin 39
2.38.1 函数说明 39
2.38.2 描述 39
2.38.3 使用例程 39
2.39 MLIB_Cos 39
2.39.1 函数说明 39
2.39.2 描述 39
2.39.3 使用例程 40
2.40 MLIB_SinCos 40
2.40.1 函数说明 40
2.40.2 描述 40
2.40.3 使用例程 40
3 修改记录 41

1 引言

1.1 概述

该用户指南描述了基于灵动微电子的 Arm® Cortex®-M0 内核 MCU 电机控制数学库(MLIB)。该库包含了取绝对值、加法、减法、乘法、除法、取反、左移、右移等函数。

* 注意:所有例程的测试时间均基于 72MHZ 主频的 MCU 且在 RAM 中执行所得。

1.2 数据类型

MLIB 支持的数据类型:(无符号)整型。整数数据类型对于通用计算非常有用。

以下列表为库中定义的整数类型:

•无符号 16 位整数—< 0；65535 >最小分辨率 1

•有符号 16 位整数—<-32768；32767 >最小分辨率 1

•无符号 32 位整数—< 0；4294967295 >最小分辨率为 1

•有符号 32 位整数—<-2147483648；2147483647 >最低分辨率为 1

以下列表为库中定义的小数类型:

•定点 16 位小数—<-1；1 - 2^{-15} >最小分辨率为 2^{-15}

•定点 32 位小数—<-1；1 - 2^{-31} >最小分辨率为 2^{-31}

1.3 例程定义

为了实现函数的简单调用，函数名使用前缀和后缀来区分函数版本，请参考以下示例:

```
s16 _Result = MLIB_Mul_Q15(s16Mult1, s16Mult2);
```

其中函数名由三部分组成:

- MLIB—库前缀
- Mul—函数功能
- Q15—定标格式

2 库函数说明

2.1 MLIB_Neg_Q15

MLIB_Neg_Q15()为没有限幅的取反函数，等式如下所示:

$$MLIB_Neg_Q15(x) = -x \quad (1)$$

2.1.1 函数说明

表 2-1 MLIB_Neg_Q15 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Neg_Q15	Q15_t	Q15_t	226ns

2.1.2 描述

对 MLIB_Neg_Q15()函数有以下声明：

```
static inline Q15_t MLIB_Neg_Q15(register Q15_t s16Val)
```

2.1.3 使用例程

以下示例为 MLIB_Neg_Q15 函数的用法：

```
1.  #include "mlib.h”
2.  Q15_t  s16_Result;
3.  Q15_t  s16Val;
4.  int main(void)
5.  {
6.     s16Val = Q15(-0.4);
7.     /*s16_Result= -s16Val = Q15(0.4)*/
8.     s16_Result = MLIB_Neg_Q15(s16Val);
9.  }
```

2.2 MLIB_Neg_Q31

MLIB_Neg_Q31()为没有限幅的取反函数，等式如下所示:

$$MLIB_Neg_Q31(x) = -x \quad (2)$$

2.2.1 函数说明

表 2-2 MLIB_Neg_Q31 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Neg_Q31	Q31_t	Q31_t	212ns

2.2.2 描述

对 MLIB_Neg_Q31 函数有以下声明：

```
static inline Q31_t MLIB_Neg_Q31(register Q31_t s32Val)
```

2.2.3 使用例程

以下示例为 MLIB_Neg_Q31 函数的用法：

```
#include "mlib.h”
Q31_t  s32_Result;
Q31_t  s32Val;
int main(void)
{
   s32Val = Q31(-0.4);
   /* s32_Result =-s32Val = Q31(0.4) */
   s32_Result = MLIB_Neg_Q31(s32Val);
}
```

2.3 MLIB_NegSat_Q31

MLIB_NegSat_Q31()为含有限幅的取反函数，等式如下所示:

$$MLIB_NegSat_Q31(x) = -x \quad (3)$$

2.3.1 函数说明

表 2-3 MLIB_NegSat_Q31 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Neg_Q31	Q31_t	Q31_t	378ns

2.3.2 描述

对 MLIB_NegSat_Q31 函数有以下声明：

```
static inline Q31_t MLIB_NegSat_Q31(register Q31_t s32Val)
```

2.3.3 使用例程

以下示例为 MLIB_NegSat_Q31 函数的用法：

```
#include "mlib.h”
Q31_t  s32_Result;
Q31_t  s32Val;
int main(void)
{
   s32Val = Q31(-0.4);
   /* s32_Result = -s32Val = Q31(0.4)*/
   s32_Result = MLIB_NegSat_Q31(s32Val);
}
```

2.4 MLIB_Abs_Q15

MLIB_Abs_Q15()为没有限幅的取绝对值函数，等式如下所示:

$$MLIB_Abs_Q15(x) = |x| \quad (4)$$

2.4.1 函数说明

表 2-4 MLIB_Abs_Q15 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Abs_Q15	Q15_t	Q15_t	294ns

2.4.2 描述

对 MLIB_Abs_Q15()函数有以下声明：

```
static inline Q15_t MLIB_Abs_Q15 (register Q15_t s16Val)
```

2.4.3 使用例程

以下示例为 MLIB_Abs_Q15 函数的用法：

```
1.  #include "mlib.h”
2.  Q15_t  s16_Result;
3.  Q15_t  s16Val;
4.  int main(void)
5.  {
6.     s16Val = Q15(-0.4);
7.     /* s16_Result = | s16Val | = Q15(0.4)*/
8.     s16_Result = MLIB_Abs_Q15(s16Val);
9.  }
```

2.5 MLIB_AbsSat_Q15

MLIB_AbsSat_Q15()为含有限幅的取绝对值函数，等式如下所示:

$$MLIB_AbsSat_Q15(x) = |x| \quad (5)$$

2.5.1 函数说明

表 2-5 MLIB_AbsSat_Q15 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_AbsSat_Q15	Q15_t	Q15_t	446ns

2.5.2 描述

对 MLIB_AbsSat_Q15()函数有以下声明：

```
static inline Q15_t MLIB_AbsSat_Q15 (register Q15_t s16Val)
```

2.5.3 使用例程

以下示例为 MLIB_AbsSat_Q15 函数的用法：

```
#include "mlib.h”
Q15_t  s16_Result;
Q15_t  s16Val;
int main(void)
{
   s16Val = Q15(-0.4);
   /* s16_Result = | s16Val | = Q15(0.4)*/
   s16_Result = MLIB_AbsSat_Q15(s16Val);
}
```

2.6 MLIB_Abs_Q31

MLIB_Abs_Q31()为没有限幅的取绝对值函数，等式如下所示:

$$\mathrm{MLIB_Abs_Q31}(x) = |x| \quad (6)$$

2.6.1 函数说明

表 2-6 MLIB_Abs_Q31 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Abs_Q31	Q31_t	Q31_t	278ns

2.6.2 描述

对 MLIB_Abs_Q31()函数有以下声明：

```
static inline Q31_t MLIB_Abs_Q31 (register Q31_t s32Val)
```

2.6.3 使用例程

以下示例为 MLIB_Abs_Q31 函数的用法：

```
#include "mlib.h”
Q31_t  s32_Result;
Q31_t  s32Val;
int main(void)
{
   s32Val = Q31(-0.4); /* s32Val = Q31(-0.4) */
   /* s32_Result = | s32Val | = Q31(0.4)*/
   s32_Result = MLIB_Abs_Q31(s32Val);
}
```

2.7 MLIB_AbsSat_Q31

MLIB_AbsSat_Q31()为含有限幅的取绝对值函数，等式如下所示:

$$MLIB_AbsSat_Q31(x) = |x| \quad (7)$$

2.7.1 函数说明

表 2-7 MLIB_AbsSat_Q31 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_AbsSat_Q31	Q31_t	Q31_t	446ns

2.7.2 描述

对 MLIB_AbsSat_Q31()函数有以下声明：

```
static inline Q31_t MLIB_AbsSat_Q31 (register Q31_t s32Val)
```

2.7.3 使用例程

以下示例为 MLIB_AbsSat_Q31 函数的用法：

```
1.  #include "mlib.h”
2.  Q31_t  s32_Result;
3.  Q31_t  s32Val;
4.  int main(void)
5.  {
6.     s32Val = Q31(-0.4);
7.     /* s32_Result = | s32Val | = Q31(0.4)*/
8.     s32_Result = MLIB_AbsSat_Q31(s32Val);
9.  }
```

2.8 MLIB_Add_Q15

MLIB_Add_Q15()没有限幅的求和函数，等式如下所示:

$$MLIB_Add_Q15(x_1,x_2) = x_1 + x_2 \quad (8)$$

2.8.1 函数说明

表 2-8 MLIB_Add_Q15 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Add_Q15	Q15_t、Q15_t	Q15_t	306ns

2.8.2 描述

对 MLIB_Add_Q15()函数有以下声明：

```
static inline Q15_t MLIB_Add_Q15(register Q15_t s16Add1, register
Q15_t s16Add2)
```

2.8.3 使用例程

以下示例为 MLIB_Add_Q15 函数的用法：

```
1.  #include "mlib.h”
2.  Q15_t  s16_Result;
3.  Q15_t  s16Add1;
4.  Q15_t  s16Add2;
5.  int main(void)
6.  {
7.     s16Add1 = Q15(0.4);
8.     s16Add2 = Q15(0.4);
9.     /* s16_Result = s16Add1 + s16Add2 = Q15(0.8)*/
10.    s16_Result = MLIB_Add_Q15(s16Add1, s16Add2);
11. }
```

2.9 MLIB_AddSat_Q15

MLIB_AddSat_Q15()含有限幅的求和函数，等式如下所示:

$$MLIB_AddSat_Q15(x_1,x_2) = x_1 + x_2 \quad (9)$$

2.9.1 函数说明

表 2-9 MLIB_AddSat_Q15 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_AddSat_Q15	Q15_t、Q15_t	Q15_t	540ns

2.9.2 描述

对 MLIB_AddSat_Q15()函数有以下声明：

```
static inline Q15_t MLIB_AddSat_Q15 (register Q15_t s16Add1, register Q15_t s16Add2)
```

2.9.3 使用例程

以下示例为 MLIB_AddSat_Q15 函数的用法：

```
1.  #include "mlib.h”
2.  Q15_t  s16_Result;
3.  Q15_t  s16Add1;
4.  Q15_t  s16Add2;
5.  int main(void)
6.  {
7.     s16Add1 = Q15(0.4);
8.     s16Add2 = Q15(0.4);
9.     /* s16_Result = s16Add1 + s16Add2 = Q15(0.8) */
10.    s16_Result = MLIB_AddSat_Q15(s16Add1, s16Add2);
11. }
```

2.10 MLIB_Add_Q31

MLIB_Add_Q31()没有限幅的求和函数，等式如下所示:

$$MLIB_Add_Q31(x_1,x_2) = x_1 + x_2 \quad (10)$$

2.10.1 函数说明

表 2-10 MLIB_Add_Q31 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Add_Q31	Q31_t、Q31_t	Q31_t	264ns

2.10.2 描述

对 MLIB_Add_Q31()函数有以下声明：

```
static inline Q31_t MLIB_Add_Q31 (register Q31_t s32Add1, register
Q31_t s32Add2)
```

2.10.3 使用例程

以下示例为 MLIB_Add_Q31 函数的用法：

```
1.  #include "mlib.h"
2.  Q31_t  s32_Result;
3.  Q31_t  s32_Add1;
4.  Q31_t  s32_Add2;
5.  int main(void)
6.  {
7.     s32_Add1 = Q31(0.4);
8.     s32_Add2 = Q31(0.4);
9.     /* s32_Result = s32_Add1 + s32_Add2 = Q31(0.8)*/
10.     s32_Result = MLIB_Add_Q31(s32_Add1, s32_Add2);
11.   }
```

2.11 MLIB_AddSat_Q31

MLIB_AddSat_Q31()含有限幅的求和函数，等式如下所示:

$$MLIB_AddSat_Q31(x_1,x_2) = x_1 + x_2 \quad (11)$$

2.11.1 函数说明

表 2-11 MLIB_AddSat_Q31 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_AddSat_Q31	Q31_t、Q31_t	Q31_t	568ns

2.11.2 描述

对 MLIB_AddSat_Q31()函数有以下声明：

```
static inline Q31_t MLIB_AddSat_Q31 (register Q31_t s32Add1, register Q31_t s32Add2)
```

2.11.3 使用例程

以下示例为 MLIB_AddSat_Q31 函数的用法：

```
1.  # include "mlib.h”
2.  Q31_t  s32_Result;
3.  Q31_t  s32_Add1;
4.  Q31_t  s32_Add2;
5.  int main(void)
6.  {
7.     s32_Add1 = Q31(0.4);
8.     s32_Add2 = Q31(0.4);
9.     /* s32 _Result = s32_Add1+ s32_Add2 = Q31(0.8)*/
10.    s32_Result = MLIB_AddSat_Q31(s32_Add1, s32_Add2);
11. }
```

2.12 MLIB_ShL_Q15

MLIB_ShL_Q15()为左移函数，函数输入有两个变量，一个为输入变量，另一个为左移的位数。

2.12.1 函数说明

表 2-12 MLIB_ShL_Q15 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_ShL_Q15	Q15_t、uint16_t	Q15_t	318ns

* 注意: 左移位数 u16Sh 的范围为<0;15>。

2.12.2 描述

对 MLIB_ShL_Q15()函数有以下声明：

```
static inline Q15_t MLIB_ShL_Q15 (register Q15_t s16Val, register uint16_t u16Sh)
```

2.12.3 使用例程

以下示例为 MLIB_ShL_Q15 函数的用法：

```
1.  # include "mlib.h”
2.  Q15_t  s16_Result;
3.  Q15_t  s16Val;
4.  uint16_t u16Sh;
5.  int main(void)
6.  {
7.     s16Val = Q15(0.1);
8.     u16Sh = 2;
```

```
9.      /* s16_Result = s16Val << u16Sh = Q15(0.4)*/
10.     s16_Result = MLIB_ShL_Q15(s16Val, u16Sh);
11.  }
```

2.13 MLIB_ShR_Q15

MLIB_ShR_Q15()为右移函数，函数输入有两个变量，一个为输入变量，另一个为右移的位数。

2.13.1 函数说明

表 2-13 MLIB_ShR_Q15 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_ShR_Q15	Q15_t、uint16_t	Q15_t	306ns

* 注意：右移位数 u16Sh 的范围为<0;15>。

2.13.2 描述

对 MLIB_ShR_Q15()函数有以下声明：

```
static inline Q15_t MLIB_ShR_Q15 (register Q15_t s16Val, register
uint16_t u16Sh)
```

2.13.3 使用例程

以下示例为 MLIB_ShR_Q15 函数的用法：

```
1.   # include "mlib.h”
2.   Q15_t  s16_Result;
3.   Q15_t  s16Val;
4.   uint16_t  u16Sh;
5.   int main(void)
6.   {
7.      s16Val = Q15(0.4);
8.      u16Sh = 2;
9.      /* s16_Result = s16Val >> u16Sh = Q15(0.1)*/
10.     s16_Result = MLIB_ShR_Q15(s16Val, u16Sh);
11.  }
```

2.14 MLIB_ShLBi_Q15

MLIB_ShLBi_Q15()为左移函数，函数输入有两个变量，一个为输入变量，另一个为左移的位数，并且函数含有双向移位功能（i16Sh 为负数，输出结果右移）。

2.14.1 函数说明

表 2-14 MLIB_ShLBi_Q15 函数说明

功能名	输入类型	输出类型	执行时间

MLIB_ShLBi_Q15	Q15_t、int16_t	Q15_t	416ns

*注意：左移位数 u16Sh 的范围为<-15;15>。

2.14.2 描述

对 MLIB_ShLBi_Q15()函数有以下声明：

```
static inline Q15_t MLIB_ShLBi_Q15 (register Q15_t s16Val, register int16_t i16Sh)
```

2.14.3 使用例程

以下示例为 MLIB_ShLBi_Q15 函数的用法：

```
1.  # include "mlib.h”
2.  Q15_t s  16_Result;
3.  Q15_t s  16Val;
4.  int16_t   i16Sh;
5.  int main(void)
6.  {
7.     s16Val = Q15(0.1);
8.     i16Sh = 2;
9.     /* s16_Result = s16Val << i16Sh = Q15(0.4)*/
10.    s16_Result = MLIB_ShLBi_Q15(s16Val, i16Sh);
11. }
```

2.15 MLIB_ShRBi_Q15

MLIB_ShRBi_Q15()为右移函数，函数输入有两个变量，一个为输入变量，另一个为右移的位数，并且函数含有双向移位功能（i16Sh 为负数，输出结果左移）。

2.15.1 函数说明

表 2-15 MLIB_ShRBi_Q15 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_ShRBi_Q15	Q15_t、int16_t	Q15_t	428ns

* 注意：右移位数 u16Sh 的范围为<-15;15>。

2.15.2 描述

对 MLIB_ShR_Q15()函数有以下声明：

```
static inline Q15_t MLIB_ShRBi_Q15 (register Q15_t s16Val, register int16_t i16Sh)
```

2.15.3 使用例程

以下示例为 MLIB_ShRBi_Q15 函数的用法：

```
1.  # include "mlib.h”
```

```
2.   Q15_t  s16_Result;
3.   Q15_t  s16Val;
4.   int16_t  i16Sh;
5.   int main(void)
6.   {
7.      s16Val = Q15(0.4);
8.      i16Sh = 2;
9.      /* s16_Result = s16Val >> i16Sh = Q15(0.1) */
10.     s16_Result = MLIB_ShRBi_Q15(s16Val, i16Sh);
11.  }
```

2.16 MLIB_Sub_Q15

MLIB_Sub_Q15()为没有限幅的减法函数，等式如下所示:

$$MLIB_Sub_Q15(x_1,x_2) = x_1 - x_2 \quad (12)$$

2.16.1 函数说明

表 2-16 MLIB_Sub_Q15 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Sub_Q15	Q15_t、Q15_t	Q15_t	306ns

2.16.2 描述

对 MLIB_Sub_Q15()函数有以下声明：

```
static inline Q15_t MLIB_Sub_Q15 (register Q15_t s16Min, register Q15_t s16Sub)
```

2.16.3 使用例程

以下示例为 MLIB_Sub_Q15 函数的用法：

```
1.    # include "mlib.h”
2.    Q15_t  s16_Result;
3.    Q15_t  s16Min;
4.    Q15_t  s16Sub;
5.    int main(void)
6.    {
7.       s16Min = Q15(0.4);
8.       s16Sub = Q15(0.3);
9.       /* s16_Result = s16Min-s16Sub = Q15(0.1)*/
10.      s16_Result = MLIB_Sub_Q15(s16Min, s16Sub);
11.   }
```

2.17 MLIB_SubSat_Q15

MLIB_SubSat_Q15()含有限幅的减法函数，等式如下所示:

$$MLIB_SubSat_Q15(x_1,x_2) = x_1 - x_2 \quad (13)$$

2.17.1 函数说明

表 2-17 MLIB_SubSat_Q15 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_SubSat_Q15	Q15_t、Q15_t	Q15_t	542ns

2.17.2 描述

对 MLIB_SubSat_Q15()函数有以下声明：

```
static inline Q15_t MLIB_SubSat_Q15 (register Q15_t s16Min, register Q15_t s16Sub)
```

2.17.3 使用例程

以下示例为 MLIB_SubSat_Q15 函数的用法：

```
1.  # include "mlib.h”
2.  Q15_t  s16_Result;
3.  Q15_t  s16Min;
4.  Q15_t  s16Sub;
5.  int main(void)
6.  {
7.     s16Min = Q15(0.4);
8.     s16Sub = Q15(0.3);
9.     /* s16_Result = s16Min-s16Sub = Q15(0.1)*/
10.    s16_Result = MLIB_SubSat_Q15(s16Min, s16Sub);
11. }
```

2.18 MLIB_Sub_Q31

MLIB_Sub_Q31()没有限幅的减法函数，等式如下所示:

$$MLIB_Sub_Q31(x_1,x_2) = x_1 - x_2 \quad (14)$$

2.18.1 函数说明

表 2-18 MLIB_Sub_Q31 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Sub_Q31	Q31_t、Q31_t	Q31_t	262ns

2.18.2 描述

对 MLIB_Sub_Q31()函数有以下声明：

```
static inline Q31_t MLIB_Sub_Q31 (register Q31_t s32Min, register Q31_t s32Sub)
```

2.18.3 使用例程

以下示例为 MLIB_Sub_Q31 函数的用法：

```
1.  # include "mlib.h”
2.  Q31_t  s32_Result;
3.  Q31_t  s32Min;
4.  Q31_t  s32Sub;
5.  int main(void)
6.  {
7.     s32Min = Q31(0.4);
8.     s32Sub = Q31(0.3);
9.     /* s32_Result = s32Min-s32Sub = Q31(0.1)*/
10.    s32_Result = MLIB_Sub_Q31(s32Min, s32Sub);
11. }
```

2.19 MLIB_SubSat_Q31

MLIB_SubSat_Q31()含有限幅的减法函数，等式如下所示:

$$MLIB_SubSat_Q31(x_1,x_2) = x_1 - x_2 \quad (15)$$

2.19.1 函数说明

表 2-19 MLIB_SubSat_Q31 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_SubSat_Q31	Q31_t、Q31_t	Q31_t	570ns

2.19.2 描述

对 MLIB_SubSat_Q31()函数有以下声明：

```
static inline Q31_t MLIB_SubSat_Q31 (register Q31_t s32Min, register
Q31_t s32Sub)
```

2.19.3 使用例程

以下示例为 MLIB_SubSat_Q31 函数的用法：

```
1.  # include "mlib.h”
2.  Q31_t  s32_Result;
3.  Q31_t  s32Min;
4.  Q31_t  s32Sub;
5.  int main(void)
6.  {
7.     s32Min = Q31(0.4);
8.     s32Sub = Q31(0.3);
9.     /* s32_Result = s32Min-s32Sub = Q31(0.1)*/
10.    s32_Result = MLIB_SubSat_Q31(s32Min, s32Sub);
12. }
```

2.20 MLIB_ShL_Q31

MLIB_ShL_Q31()为左移函数，函数输入有两个变量，一个为输入变量，另一个为左移的位数。

2.20.1 函数说明

表 2-20 MLIB_ShL_Q31 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_ShL_Q31	Q31_t、uint16_t	Q31_t	292ns

* 注意: 左移位数 u16Sh 的范围为<0;31>。

2.20.2 描述

对 MLIB_ShL_Q31()函数有以下声明：

```
static inline Q31_t MLIB_ShL_Q31 (register Q31_t s32Val, register
uint16_t u16Sh)
```

2.20.3 使用例程

以下示例为 MLIB_ShL_Q31 函数的用法：

```
1.   # include "mlib.h”
2.   Q31_t  s32_Result;
3.   Q31_t  s32Val;
4.   uint16_t  u16Sh;
5.   int main(void)
6.   {
7.      s32Val = Q31(0.1);
8.      u16Sh = 2;
9.      /* s32_Result = s32Val << u16Sh = Q31(0.4) */
10.     s32_Result = MLIB_ShL_Q31(s32Val, u16Sh);
11.  }
```

2.21 MLIB_ShR_Q31

MLIB_ShR_Q31()为右移函数，函数输入有两个变量，一个为输入变量，另一个为右移的位数。

2.21.1 函数说明

表 2-21 MLIB_ShR_Q31 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_ShR_Q31	Q31_t、uint16_t	Q31_t	294ns

* 注意: 右移位数 u16Sh 的范围为<0;31>。

2.21.2 描述

对 MLIB_ShR_Q31()函数有以下声明：

```
static inline Q31_t MLIB_ShR_Q31 (register Q31_t s32Val, register
uint16_t u16Sh)
```

2.21.3 使用例程

以下示例为 MLIB_ShR_Q31 函数的用法：

```
1.   # include "mlib.h”
2.   Q31_t  s32_Result;
3.   Q31_t  s32Val;
4.   uint16_t  u16Sh;
5.   int main(void)
6.   {
7.      s32Val = Q31(0.4);
8.      u16Sh = 2;
9.      /* s32_Result = s32Val >> u16Sh = Q31(0.1)*/
10.     s32_Result = MLIB_ShR_Q31(s32Val, u16Sh);
11.  }
```

2.22 MLIB_ShLBi_Q31

MLIB_ShLBi_Q31()为左移函数，函数输入有两个变量，一个为输入变量，另一个为左移的位数，并且函数含有双向移位功能（i16Sh 为负数，输出结果右移）。

2.22.1 函数说明

表 2-22 MLIB_ShLBi_Q31 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_ShLBi_Q31	Q31_t、int16_t	Q31_t	402ns

* 注意：左移位数 u16Sh 的范围为<-31;31>。

2.22.2 描述

对 MLIB_ShLBi_Q31()函数有以下声明：

```
static inline Q31_t MLIB_ShLBi_Q31 (register Q31_t s32Val, register
int16_t i16Sh)
```

2.22.3 使用例程

以下示例为 MLIB_ShLBi_Q31 函数的用法：

```
1.   # include "mlib.h”
2.   Q31_t  s32_Result;
3.   Q31_t  s32Val;
4.   int16_t  i16Sh;
5.   int main(void)
```

```
{
   s32Val = Q31(0.1);
   i16Sh = 2;
   /*s32_Result = s32Val << i16Sh = Q31(0.4) *
   s32_Result = MLIB_ShLBi_Q31(s32Val, i16Sh);
}
```

2.23 MLIB_ShRBi_Q31

MLIB_ShRBi_Q31()为右移函数，函数有两个变量，一个为输入变量，另一个为右移的位数，并且函数含有双向移位功能（i16Sh 为负数，输出结果左移）。

2.23.1 函数说明

表 2-23 MLIB_ShRBi_Q31 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_ShRBi_Q31	Q31_t、int16_t	Q31_t	402ns

* 注意：右移位数 u16Sh 的范围为<-31;31>。

2.23.2 描述

对 MLIB_ShRBi_Q31()函数有以下声明：

```
static inline Q31_t MLIB_ShRBi_Q31 (register Q31_t s32Val, register
int16_t i16Sh)
```

2.23.3 使用例程

以下示例为 MLIB_ShRBi_Q31 函数的用法：

```
# include "mlib.h”
Q31_t  s32_Result;
Q31_t  s32Val;
int16_t  i16Sh;
int main(void)
{
   s32Val = Q31(0.4);
   i16Sh = 2;
   /*s32_Result = s32Val>> i16Sh = Q31(0.1) *
   s32_Result = MLIB_ShRBi_Q31(s32Val, i16Sh);
}
```

2.24 MLIB_ShLSat_Q31

MLIB_ShLSat_Q31()为含有限幅的左移函数，函数有两个变量，一个为输入变量，另一个为左移的位数。

2.24.1 函数说明

表 2-24 MLIB_ShLSat_Q31 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_ShLSat_Q31	Q31_t、uint16_t	Q31_t	470ns

* 注意：左移位数 u16Sh 的范围为<0;31>。

2.24.2 描述

对 MLIB_ShLSat_Q31()函数有以下声明：

```
static inline Q31_t MLIB_ShLSat_Q31 (register Q31_t s32Val, register
uint16_t u16Sh)
```

2.24.3 使用例程

以下示例为 MLIB_ShLSat_Q31 函数的用法：

```
1.  # include "mlib.h”
2.  Q31_t  s32_Result;
3.  Q31_t  s32Val;
4.  uint16_t  u16Sh;
5.  int main(void)
6.  {
7.     s32Val = Q31(0.1);
8.     u16Sh = 2;
9.     /*s32_Result = s32Val << u16Sh = Q31(0.4) *
10.    s32_Result = MLIB_ShLSat_Q31(s32Val, u16Sh);
11. }
```

2.25 MLIB_Conv_Q31s

MLIB_Conv_Q31s()函数是将 16 位的数转化为 32 位的数。

2.25.1 函数说明

表 2-25 MLIB_Conv_Q31s 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Conv_Q31s	Q15_t	Q31_t	220ns

2.25.2 描述

对 MLIB_Conv_Q31s()函数有以下声明：

```
static inline Q31_t MLIB_Conv_Q31s (register Q15_t s16Val)
```

2.25.3 使用例程

以下示例为 MLIB_Conv_Q31s 函数的用法：

```
1.  # include "mlib.h”
2.  Q31_t  s32_Result;
3.  Q15_t  s16Val;
```

```
4.   int main(void)
5.   {
6.      s16Val = Q15(0.4);
7.      /* s32_Result = s16Val = Q31(0.4) */
8.      s32_Result = MLIB_Conv_Q31s(s16Val);
9.   }
```

2.26 MLIB_Conv_Q15s

MLIB_Conv_Q15s()函数是将 32 位的数转化为 16 位的数。

2.26.1 函数说明

表 2-26 MLIB_Conv_Q15s 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Conv_Q15s	Q31_t	Q15_t	208ns

2.26.2 描述

对 MLIB_Conv_Q15s()函数有以下声明：

```
static inline Q15_t MLIB_Conv_Q15s(register Q31_t s32Val)
```

2.26.3 使用例程

以下示例为 MLIB_Conv_Q15s 函数的用法：

```
1.   # include "mlib.h”
2.   Q15_t  s16_Result;
3.   Q31_t  s32Val;
4.   int main(void)
5.   {
6.      s32Val = Q31(0.4);
7.      /* s16_Result = s32Val = Q15(0.4)*/
8.      s16_Result = MLIB_Conv_Q15s(s32Val);
9.   }
```

2.27 MLIB_Mul_Q15

MLIB_Mul_Q15()为没有限幅的乘法函数，等式如下所示:

$$MLIB_Mul_Q15(x_1,x_2) = x_1 \bullet x_2 \quad (16)$$

2.27.1 函数说明

表 2-27 MLIB_Mul_Q15 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Mul_Q15	Q15_t、Q15_t	Q15_t	332ns

2.27.2 描述

对 MLIB_Mul_Q15()函数有以下声明：

```
static inline Q15_t MLIB_Mul_Q15 (register Q15_t s16Mult1, register
Q15_t s16Mult2)
```

2.27.3 使用例程

以下示例为 MLIB_Mul_Q15 函数的用法：

```
1.  # include "mlib.h”
2.  Q15_t  s16_Result;
3.  Q15_t  s16Mult1;
4.  Q15_t  s16Mult2;
5.  int main(void)
6.  {
7.     s16Mult1 = Q15(0.5);
8.     s16Mult2 = Q15(0.5);
9.     /* s16_Result = s16Mult1* s16Mult2 = Q15(0.25) */
10.    s16_Result = MLIB_Mul_Q15(s16Mult1, s16Mult2);
11. }
```

2.28 MLIB_Mul_Q31

MLIB_Mul_Q31()为没有限幅的乘法函数，等式如下所示:

$$MLIB_Mul_Q31(x_1,x_2) = x_1 \bullet x_2 \quad (17)$$

2.28.1 函数说明

表 2-28 MLIB_Mul_Q31 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Mul_Q31	Q31_t、Q31_t	Q31_t	444ns

2.28.2 描述

对 MLIB_Mul_Q31()函数有以下声明：

```
static inline Q31_t MLIB_Mul_Q31(register Q31_t s32Mult1, register Q31_t
s32Mult2)
```

2.28.3 使用例程

以下示例为 MLIB_Mul_Q31 函数的用法：

```
1.  # include "mlib.h”
2.  Q31_t  s32_Result;
3.  Q31_t  s32Mult1;
4.  Q31_t  s32Mult2;
5.  int main(void)
6.  {
```

```
7.      s32Mult1 = Q31(0.5);
8.      s32Mult2 = Q31(0.5);
9.      /* s32_Result = s32Mult1* s32Mult2 = Q31(0.25)*/
10.     s32_Result = MLIB_Mul_Q31(s32Mult1, s32Mult2);
11.  }
```

2.29 MLIB_Mul_Q31ss

MLIB_Mul_Q31ss()为没有限幅的乘法函数，等式如下所示:

$$MLIB_Mul_Q31ss(x_1,x_2) = x_1 \bullet x_2 \quad (18)$$

2.29.1 函数说明

表 2-29 MLIB_Mul_Q31ss 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Mul_Q31ss	Q15_t、Q15_t	Q31_t	320ns

2.29.2 描述

对 MLIB_Mul_Q31ss()函数有以下声明：

```
static inline Q31_t MLIB_Mul_Q31ss (register Q15_t s16Mult1, register
Q15_t s16Mult2)
```

2.29.3 使用例程

以下示例为 MLIB_Mul_Q31ss 函数的用法：

```
1.   # include "mlib.h”
2.   Q31_t  s32_Result;
3.   Q15_t  s16Mult1;
4.   Q15_t  s16Mult2;
5.   int main(void)
6.   {
7.      s16Mult1 = Q15(0.5);
8.      s16Mult2 = Q15(0.5);
9.      /* s32_Result = s16Mult1* s16Mult2 = Q31(0.25)*/
10.     s32_Result = MLIB_Mul_Q31ss(s16Mult1, s16Mult2);
11.  }
```

2.30 MLIB_MulSat_Q31

MLIB_MulSat_Q31()为含有限幅的乘法函数，等式如下所示:

$$MLIB_MulSat_Q31(x_1,x_2) = x_1 \bullet x_2 \quad (19)$$

2.30.1 函数说明

表 2-30 MLIB_MulSat_Q31 函数说明

功能名	输入类型	输出类型	执行时间

MLIB_MulSat_Q31	Q31_t、Q31_t	Q31_t	596ns

2.30.2 描述

对 MLIB_MulSat_Q31()函数有以下声明：

```
static inline Q31_t MLIB_MulSat_Q31 (register Q31_t s32Mult1, register
Q31_t s32Mult2)
```

2.30.3 使用例程

以下示例为 MLIB_MulSat_Q31 函数的用法：

```
1.  # include "mlib.h”
2.  Q31_t  s32_Result;
3.  Q31_t  s32Mult1;
4.  Q31_t  s32Mult2;
5.  int main(void)
6.  {
7.     s32Mult1 = Q31(0.5);
8.     s32Mult2 = Q31(0.5);
9.     /* s32_Result = s32Mult1* s32Mult2 = Q31(0.25)*/
10.    s32_Result = MLIB_MulSat_Q31(s32Mult1, s32Mult2);
11. }
```

2.31 MLIB_MulSat_Q31ss

MLIB_MulSat_Q31ss()为含有限幅的乘法函数，等式如下所示:

$$MLIB_MulSat_Q31ss(x_1,x_2) = x_1 \bullet x_2 \quad (20)$$

2.31.1 函数说明

表 2-31 MLIB_MulSat_Q31ss 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_MulSat_Q31ss	Q15_t 、Q15_t	Q31_t	472ns

2.31.2 描述

对 MLIB_MulSat_Q31ss()函数有以下声明：

```
static inline Q31_t MLIB_MulSat_Q31ss (register Q15_t s16Mult1, register
Q15_t s16Mult2)
```

2.31.3 使用例程

以下示例为 MLIB_MulSat_Q31ss 函数的用法：

```
1.  # include "mlib.h”
2.  Q31_t  s32_Result;
3.  Q15_t  s16Mult1;
```

```
Q15_t  s16Mult2;
int main(void)
{
   s16Mult1 = Q15(0.5);
   s16Mult2 = Q15(0.5);
   /* s32_Result = s16Mult1* s16Mult2 = Q31(0.25)*/
   s32_Result = MLIB_MulSat_Q31ss(s16Mult1, s16Mult2);
}
```

2.32 MLIB_Mul_Q31s

MLIB_Mul_Q31s()为没有限幅的乘法函数，等式如下所示:

$$MLIB_Mul_Q31s(x_1,x_2) = x_1 \cdot x_2 \quad (21)$$

2.32.1 函数说明

表 2-32 MLIB_Mul_Q31s 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Mul_Q31s	Q31_t 、Q15_t	Q31_t	360ns

2.32.2 描述

对 MLIB_Mul_Q31()函数有以下声明：

```
static inline Q31_t MLIB_Mul_Q31s (register Q31_t s32Mult1, register
Q15_t s16Mult2)
```

2.32.3 使用例程

以下示例为 MLIB_Mul_Q31s 函数的用法：

```
# include "mlib.h”
Q31_t  s32_Result;
Q31_t  s32Mult1;
Q15_t  s16Mult2;
int main(void)
{
   s32Mult1 = Q31(0.5);
   s16Mult2 = Q15(0.5);
   /* s32_Result = s32Mult1* s16Mult2 = Q31(0.25) */
   s32_Result = MLIB_Mul_Q31s(s32Mult1, s16Mult2);
}
```

2.33 MLIB_FilterMA_Q15

MLIB_FilterMA_Q15 为均值滤波函数，此函数由计算平均值的递归形式得到，它可以使得输出更加平滑，滤波深度通过 MLIB_FILTER_MA_T 结构体赋值来定义的，其中 *u16Sh* 为滤波器深度，存储在滤波器的结构体中。该数字表示滤波点数为 2 的幂次方，如 *u16Sh* 为 3，函数执行深度为 2^3 次即深度为 8 的均值滤波，

如下所示：

$$n_p = 2^{(u16Sh)}, \quad u16Sh \geq 0 \qquad (22)$$

2.33.1 函数说明

表 2-33 MLIB_FilterMA_Q15 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_FilterMA_Q15	Q15_t、MLIB_FILTER_MA_T	Q15_t	1916ns

表 2-34 结构体数据类型说明

结构体	变量	输入/输出	数据类型	描述
MLIB_FILTER_MA_T	s32Acc	输入	int32_t	内部滤波器累加值
	u16Sh	输入	uint16_t	滤波器深度

2.33.2 描述

对 MLIB_FilterMA_Q15 函数有以下声明：

```
static inline Q15_t MLIB_FilterMA_Q15 (Q15_t s16InX, MLIB_FILTER_MA_T
*psParam)
```

2.33.3 使用例程

以下示例为 MLIB_FilterMA_Q15 函数的用法：

```
#include "mlib.h”
int16_t  s16Variable_in;
int16_t  s16Variable_flt;
MLIB_FILTER_MA_T s16Variable_Filt;
int main(void)
{
   s16Variable_Filt. s32Acc = 0;
   /* Filter depth 2^3 = 8 points */
   s16Variable_Filt. u16Sh= 3;
}
Void ISR(void)
{
   s16Variable_flt = MLIB_FilterMA_Q15(Variable_in  \
                   , &s16Variable_Filt);
}
```

2.34 MFLIB_SMO

MLIB_SMO 函数为滑模观测器函数，可以用来估计电机的感应电动势、速度以及位置，并且对控制对象的参数变化及扰动具有自适应能力。

我们可以使用一个 RL 负载模型来近似 PMSM，如式 23 所示，其中e_s为电机反电势、i_s为电机电流矢量、v_s为输入电压矢量、R为绕组电阻、L为绕组电感。

$$v_s = Ri_s + L\frac{d}{dt}i_s + e_s \quad (23)$$

求解i_s可得：

$$\frac{d}{dt}i_s = (-\frac{R}{L})i_s + \frac{1}{L}(v_s - e_s) \quad (24)$$

经过离散化可得：

$$i_s(n+1) = (1 - T_s \cdot \frac{R}{L})i_s(n) + \frac{T_s}{L}(v_s(n) - e_s(n)) \quad (25)$$

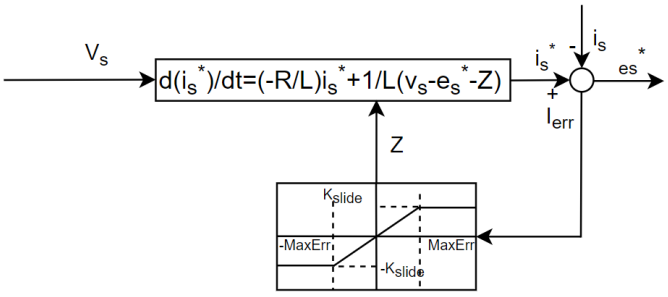
该电机模型有两个参数，需要进行修改以用于指定的电机。这两个参数分别为 F 增益和 G 增益，其中：

$$F = 1 - T_s \cdot \frac{R}{L} \quad (26)$$

$$G = \frac{T_s}{L} \quad (27)$$

其中常量 R 和 L 可以使用 LCR 表即可测量出。例如，如果测量的是线间电阻，由于在计算 F 和 G 增益时使用的是相电阻，因此需要将测量值除以 2 后代入公式。电感常量 L 的计算方法相同。SMO 的原理图如下图所示。

图 2-1 SMO 原理框图



Vs 为输入电压、is*为估算电流，es*为反电动势，Z 为校正因子，eα 和 eβ 值（es 的矢量分量）用于估算 θ*。

2.34.1 函数说明

表 2-35 MLIB_SMO 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_SMO	SMOType	None	2204ns

表 2-36 结构体数据类型说明

结构体	变量	输入/输出	数据类型	描述
SMOType	s16Fs	输入	int16_t	F 增益
	s16Gs	输入	int16_t	G 增益
	s16ErrMax	输入	int16_t	电流误差横轴限幅
	s16Kslide	输入	int16_t	电流误差纵轴限幅
	s16IerrGain	输入	int16_t	电流误差增益
	i16IerrGainShift	输入	int16_t	误差增益移位数
	s16Ialpha_est	输出	int16_t	估算得到的 α 轴电流
	s16Ibeta_est	输出	int16_t	估算得到的 β 轴电流
	s16Ialpha	输入	int16_t	α 轴电流输入
	s16Ibeta	输入	int16_t	β 轴电流输入
	s16Ealpha_est	输出	int16_t	α 轴反电势输出
	s16Ebeta_est	输出	int16_t	β 轴反电势输出
	s16Valpha	输入	int16_t	α 轴电压输入
	s16Vbeta	输入	int16_t	β 轴电压输入
	s16Z_alpha	输出	int16_t	α 轴校正因子
	s16Z_beta	输出	int16_t	β 轴校正因子
	s16Speed	输入	int16_t	电机机械速度

2.34.2 描述

对 MLIB_SMO ()函数有以下声明：

```
static inline void MLIB_SMO (SMOType *sSMOObs);
```

2.34.3 使用例程

以下示例为 MLIB_SMO 函数的用法：

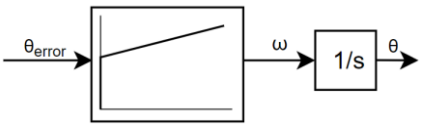
```
1.  #include "mlib.h”
2.  SMOType  SMO;
3.  int main(void)
4.  {
5.     SMO. s16Fs=Q15(0.97);
6.     SMO. s16Gs= Q15(0.06);
7.     SMO. s16ErrMax =Q15(0.3);
8.     SMO. s16Kslide =Q15(0.3);
9.     SMO. s16IerrGain = Q15(0.5);
10.    SMO. i16IerrGainShift =1;
11. }
```

```
12.  Void ISR(void)
13.  {
14.     MLIB_SMO (&SMO);
15.  }
```

2.35 MLIB_PLL

MLIB_PLL 函数为锁相环函数，其将滑模观测器算法的输出作为锁相环的输入，从而来获取转子的位置和速度，原理图如下图所示:

图 2-2 PLL 原理框图



所描述的锁相环具有以下传递函数：

$$\frac{\theta^*(s)}{\theta(s)} = \frac{sK_P + K_I}{s^2 + sK_P + K_I} \quad (28)$$

控制器增益 K_P 和 K_I 的计算方法是将得到的传递函数的特征多项式与标准二阶系统多项式进行比较。

根据图 2-2 中的方框图，实现锁相环的基本方程如下：

$$\omega(k) = K_P \cdot e(k) + T_S \cdot K_I \cdot e(k) + \omega(k-1) \quad (29)$$

$$\theta(k) = T_S \cdot \omega(k) + \theta(k-1) \quad (30)$$

式中：

K_P 为比例增益、K_I 为积分增益、T_S 为采样周期、$e(k)$ 为第 k 次的位置误差、$\omega(k)$ 为第 k 次的转子转速、$\omega(k-1)$ 为 k-1 次的转子转速、$\theta(k)$ 为第 k 次的转子脚位置、$\theta(k-1)$ 为 k-1 次的转子位置。

在浮点运算中，公式 30 和公式 31 变换如下：

$$\omega_{SC}(k)\ \omega_{max} = K_P \cdot e_{SC}(k) + T_S \cdot K_I \cdot e_{SC}(k) + \omega_{SC}(k-1)\ \omega_{max} \quad (31)$$

$$\theta_{SC}(k)\ \theta_{max} = T_S \cdot \omega_{SC}(k)\ \omega_{max} + \theta_{SC}(k-1)\ \theta_{max} \quad (32)$$

式中：

$e_{SC}(k)$ 为第 k 次的位置误差、$\omega_{SC}(k)$ 为第 k 次的转子转速、$\omega_{SC}(k-1)$ 为 k-1 次的转子转速、θ_{SC} 为第 k 次的转子位置、$\theta_{SC}(k-1)$ 为 k-1 次的转子位置、ω_{max} 为最大转速、θ_{max} 为最大转子位置。

因此，PLL_TypeDef 结构体中包含的 s16PGain、s16IGain 和 s16ThGain 参数，i16PgainSh 为 s16PGain 的移位数、i16IGainSh 为 s16IGain 的移位数、i16ThGainSh 为 s16ThGain 的移位数，s16PGain、s16IGain 和 s16ThGain 参数计算公式分别如下：

$$s16PGain = K_p \cdot \frac{1}{\omega_{max}} \quad (33)$$

$$s16IGain = T_s \cdot K_I \cdot \frac{1}{\omega_{\max}} \quad (34)$$

$$s16ThGain = T_s \frac{\omega_{\max}}{\theta_{\max}} \quad (35)$$

其中

$$K_p = 2\xi\omega_0 \quad (36)$$

$$K_I = \omega_0^2 \quad (37)$$

ω_0为截止频率，建议取$1/10\omega_N$、ξ为阻尼系数，值在 0-1 之间，建议取 0.9、ω_N为额定频率。

2.35.1 函数说明

表 2-37 MLIB_PLL 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_PLL	Q15_t PLL_TypeDef	Q15_t	1760ns

表 2-38 结构体数据类型说明

结构体	变量	输入/输出	数据类型	描述
PLL_TypeDef	s32Theta	输出	Int32_t	估算的位置
	s32Speed	输出	Int32_t	估算的速度
	s32I_1	输入	Int32_t	锁相环控制部分状态变量
	s16IGain	输入	int16_t	锁相环积分增益
	i16IGainSh	输入	int16_t	锁相环积分增益移位数
	s16PGain	输入	int16_t	锁相环比例增益
	i16PGainSh	输入	int16_t	锁相环比例增益移位数
	s16ThGain	输入	int16_t	位置输出积分增益
	i16ThGainSh	输入	int16_t	位置输出积分增益移位数

2.35.2 描述

对 MLIB_PLL ()函数有以下声明：

```
static inline Q15_t MLIB_PLL (Q15_t s16ThetaErr, PLL_TypeDef *sTO)
```

2.35.3 使用例程

以下示例为 MLIB_PLL 函数的用法：

```
1.  #include "mlib.h”
2.  PLL_TypeDef  sPLL;
3.  int16_t  s16PosElEst;
4.  int16_t  s16ThetaErr;
5.  int main(void)
6.  {
7.     sPLL.s16PGain = Q15(0.6528);
8.     sPLL.i16PGainSh=1;
9.     sPLL.s16IGain=Q15(0.988);
10.    sPLL.i16IGainSh=-9;
11.    sPLL.s16ThGain=Q15(0.666667);
12.    sPLL.i16ThGainSh=-3;
13.    sPLL.s32I_1=0;
14.    sPLL.s32Speed=0;
15. }
16. Void ISR(void)
17. {
18.    s16PosElEst=MLIB_PLL (s16ThetaErr, &sPLL);
19. }
```

2.36 MLIB_PIControl_Q15

MLIB_ PIControl_Q15 为位置式 PI 控制函数，其公式为：

$$u(t) = K_p e(t) + \frac{K_I}{T_I}\int_0^{\tau} e(t)d(t) \qquad (38)$$

$e(t)$ 为连续时域中的输入误差，由 PI 算法的 P 和 I 共同处理、$u(t)$ 为连续时域中的控制器输出、T_I 为积分时间常数[s]。

PI 算法的离散时域表示为：

$$u(k) = K_P \cdot e(k) + K_I \cdot \sum_{i=0}^{k} e(i) \qquad (39)$$

2.36.1 函数说明

表 2-39 MLIB_PIControl_Q15 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_ PIControl_Q15	Q15_t、MLIB_PIControl_T	Q15_t	2548ns

表 2-40 结构体数据类型说明

结构体	变量	输入/输出	数据类型	描述
MLIB_PIControl_T	s16Kp	输入	int16_t	比例增益

	i16Kp_Shift	输入	int16_t	比例移位数
	s16Ki	输入	int16_t	积分增益
	i16Ki_Shift	输入	int16_t	积分移位数
	s32IAccK_1	输出	int32_t	积分累积量
	s16UpperLim	输入	Q15_t	PI 输出上限
	s16LowerLim	输入	Q15_t	PI 输出下限
	bLimFlag	输出	bool_t	限幅标志位

2.36.2 描述

对 MLIB_ PIControl_Q15 ()函数有以下声明：

```
static inline Q15_t MLIB_PIControl_Q15 (Q15_t s16InErr, MLIB_PIControl_T
*psParam)
```

2.36.3 使用例程

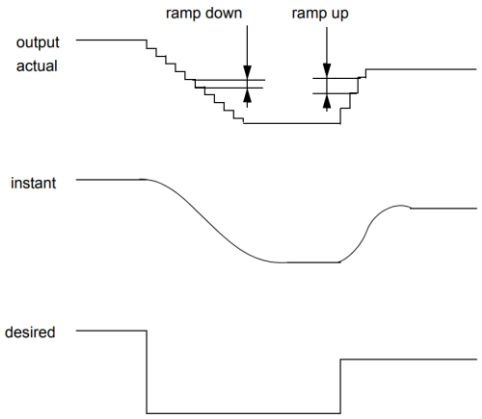
以下示例为 MLIB_ PIControl_Q15 函数的用法：

```
1.  #include "mlib.h”
2.  MLIB_PIControl_T  sIq_PI;
3.  Q15_t  s16InErr;
4.  Q15_t  s16Q ;
5.  int main(void)
6.  {
7.     sIq_PI.s16Kp=Q15(0.5);
8.     sIq_PI.s16Ki=Q15(0.03);
9.     sIq_PI.i16Kp_Shift =1;
10.    sIq_PI.i16Ki_Shift =0;
11.    sIq_PI.s16LowerLim =Q15(-0.7);
12.    sIq_PI.s16UpperLim =Q15(0.9);
13.    sIq_PI.s32IAccK_1 = 0;
14. }
15. Void ISR(void)
16. {
17.    s16Q = MLIB_PIControl_Q15(s16InErr, &sIq_PI);
18. }
```

2.37 MLIB_Ramp_Q15

MLIB_ Ramp_Q15 ()为斜坡函数，该函数通过 MLIB_RAMP_T 结构体中包含的向上或向下增量计算实际值的斜坡。如果所需值大于实际值，则函数会将上升值与实际值相加，且输出不能大于所需值。如果所需值低于实际值，该函数将从实际值中减去下降值且输出不能低于所需值，下图为具体的算法图：

图 2-3 斜坡函数算法图



2.37.1 函数说明

表 2-41 MLIB_ Ramp_Q15 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_ Ramp_Q15	Q15_t、MLIB_RAMP_T	Q15_t	1928ns

表 2-42 结构体数据类型说明

结构体	变量	输入/输出	数据类型	描述
MLIB_RAMP_T	s16RampUp	输入	Q15_t	上升斜率
	s16RampDown	输入	Q15_t	下降斜率
	s16State	输出	Q15_t	实际输出值
	bReachFlag	输出	bool_t	斜坡到达标志位

2.37.2 描述

对 MLIB_ Ramp_Q15 ()函数有以下声明：

```
static inline Q15_t MLIB_Ramp_Q15 (Q15_t s16Target, MLIB_RAMP_T *psParam);
```

2.37.3 使用例程

以下示例为 MLIB_ Ramp_Q15 函数的用法：

```
1.  #include "mlib.h”
2.  MLIB_RAMP_T  sSpeedRamp;
3.  Q15_t   s16SpdCmd;
4.  Q15_t   s16SpeedRamp;
5.  int main(void)
6.  {
7.     sSpeedRamp.bReachFlag =0;
8.     sSpeedRamp.s16RampDown =Q15(0.01);
9.     sSpeedRamp.s16RampUp =Q15(0.01);
10.    s16SpdCmd =Q15(0.9);
```

```
}
Void ISR(void)
{
   s16SpeedRamp = MLIB_Ramp_Q15(s16SpdCmd, &sSpeedRamp);
}
```

2.38 MLIB_Sin

MLIB_Sin()为正弦函数。

2.38.1 函数说明

表 2-43 MLIB_Sin 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Sin	Q15_t	Q15_t	928ns

2.38.2 描述

对 MLIB_Sin()函数有以下声明：

```
static inline Q15_t MLIB_Sin(Q15_t s16Angle)
```

2.38.3 使用例程

以下示例为 MLIB_Sin 函数的用法：

```
#include "mlib.h”
Q15_t  s16_Result;
Q15_t  s16Angle;
int main(void)
{
   s16Angle = Q15(0.333344); /* Angle = 60°*/
   /* s16_Result = sin(s16Angle) = Q15(0.863953)*/
   s16_Result = MLIB_Sin(s16Angle);
}
```

2.39 MLIB_Cos

MLIB_Cos()为余弦函数

2.39.1 函数说明

表 2-44 MLIB_Cos 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_Cos	Q15_t	Q15_t	912ns

2.39.2 描述

对 MLIB_Cos()函数有以下声明：

```
static inline Q15_t MLIB_Cos(Q15_t s16Angle)
```

2.39.3 使用例程

以下示例为 MLIB_Cos 函数的用法：

```
#include "mlib.h"
Q15_t  s16_Result;
Q15_t  s16Angle;
int main(void)
{
   s16Angle = Q15(0.333344); /* Angle = 60° */
   /* s16_Result = cos(s16Angle) = Q15(0.498199)*/
   s16_Result = MLIB_Cos(s16Angle);
}
```

2.40 MLIB_SinCos

MLIB_SinCos()为正弦余弦函数，其输出结果为同一角度的正弦值和余弦值。

2.40.1 函数说明

表 2-45 MLIB_SinCos 函数说明

功能名	输入类型	输出类型	执行时间
MLIB_SinCos	Q15_t	sAngle_Trig	1120ns

表 2-46 结构体数据类型说明

结构体	变量	输入/输出	数据类型	描述
sAngle_Trig	s16Cos	输出	Q15_t	余弦值
	s16Sin	输出	Q15_t	正弦值

2.40.2 描述

对 MLIB_SinCos()函数有以下声明：

```
static inline sAngle_Trig MLIB_SinCos(Q15_t s16Angle)
```

2.40.3 使用例程

以下示例为 MLIB_SinCos 函数的用法：

```
#include "mlib.h"
sAngle_Trig  sAngle_cmd;
Q15_t  s16Anglel;
int main(void)
{
   s16Val = Q15(0.333344); /* Val = Angle= 60°*/
   /* sAngle_cmd.s16Cos = cos(s16Angle) = Q15(0.498199) /*
   /* sAngle_cmd.s16Sin = sin(s16Angle) = Q15(0.863953) /*
   sAngle_cmd = MLIB_SinCos(s16Angle);
}
```

3 修改记录

表 3-1 修改记录

日期	版本	内容
2022/05/13	1.0	AN6001 初始版本发布