HA系列芯片C语言

仿真器用户手册

（v0.0）

目 录

[一、架构设计： 2](#_Toc45194489)

[二、CPU模块 3](#_Toc45194490)

[1. 资源 3](#_Toc45194491)

[2. 规则 3](#_Toc45194492)

[三、MEMORY 5](#_Toc45194493)

[1. 资源 5](#_Toc45194494)

[2. 规则 5](#_Toc45194495)

[四、DSP 7](#_Toc45194496)

[五、IO 10](#_Toc45194497)

[六、统计信息 10](#_Toc45194498)

[七、图形化调试 10](#_Toc45194499)

[1. memory.h 11](#_Toc45194500)

[2. CData\_io.h 13](#_Toc45194501)

[3. alu.h 15](#_Toc45194502)

[4. FMT\_F.h 20](#_Toc45194503)

[5. mac.h 21](#_Toc45194504)

[6. Math\_F.h 28](#_Toc45194505)

[7. SOC\_Common\_F.h 30](#_Toc45194506)

[8. STA\_F.h 31](#_Toc45194507)

版本记录

|  |  |  |
| --- | --- | --- |
| 版本号 | 日期 | 修改 |
| V0.0 | 2020-07-09 | 初稿 |

## 一、架构设计：



## 二、CPU模块

将CPU抽象为资源和行为的集合。资源体现为数量有限的寄存器组。行为体现为对寄存器的读写、运算、条件判断等操作的规则约束。

### 1. 资源

数据寄存器：RD0，RD1，RD2~RD31

地址寄存器：RA0，RA1，RA2，RSP，RA4~7

### 2. 规则

1> 函数的申明与调用采取C语言的标准做法。函数中需注意对寄存器的保护。

2> 原则上需遵守下表。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 操作对象  操作类型 | 寄存器 | | | | 存储器  M[…] |
| 数据寄存器RD  (RD23～RD0) | 通用地址寄存器RA  (RA2～RA0) | 专用地址寄存器RA  (RA7～RA4) | 栈指针RSP |
| 立即数操作 | RD15~0 = N (N为32bit)  RD15~0 ±= N (0≤N<256)  RD23~0 = 0 troble  RD23~0 = -1 | RA2~0 ±= N  (0≤N<256) |  |  |  |
| 数据传送 | RD1~0 <=> RD23~0  RD1~0 <=> RA7~0  RD23~0 <=> M[…]  RD1/RD0 <=> RDx  RD31~0 <=> RDx | RA2~0 <=> RD0  RA2~0 <=> RD1  RA2~0 <=> M[…]  RA2~0 = RSP | RA7~4<=>RD0  RA7~4<=>RD1  RA7~4 = RSP  RA7~4<=>M[…] | RSP <=> RD0  RSP <=> RD1  RSP <=> M[…] | M[…] <=> RD23~0  M[…] <=> RA7~0  M[…] <=> RSP |
| 单操作数指令 | Rf(RD23~0) | Rf(RA7~4)  Rf(RA2~0) | |  | Rf(M[…]) |
| 双操作数指令 | RD23~0 **Rff=** (RD0/RD1/ M[…])  M[…] **Rff=** (RD1~0)  RDx <=> RD31~0  **Rff**是 “+、-、& ”等操作符  **Rff**= 是“+=、-=、&=”等操作符  见“双操作数指令RFF” | RA7~4 **Rff=** (RD1~0)  RA2~0 **Rff=** (RD1~0) | |  |  |
| 堆栈操作 | Push RD23~0  Pop RD23~0 | Push RA2~0  Pop RA2~0 | Push RA7~4  Pop RA7~4 |  |  |
| 位操作 | RD0\_SetBit31~0  RD0\_ClrBit31~0 | | | | |

## 三、MEMORY

本模块负责跟内存相关的一切操作。如BASE\_RAM、GRAM、XRAM的空间大小的设定；堆栈指针RSP的管理；临时变量的申请；CPU对RAM的读、写操作等。

在嵌入式系统中，内存是受限资源，严格要求应用程序中不使用C语言原生的数组申请、变量申请，而改为使用本模块所提供的相关函数。

### 资源

*以320D为例：*

|  |  |
| --- | --- |
| RAM类型 | 长度(DWORD) |
| BASE\_RAM | 1 \* 1024 |
| GRAM | 16 \* 256 |
| XRAM | 8 \* 256 |

### 规则

|  |  |  |
| --- | --- | --- |
| 操作 | 汇编指令 | 模拟器函数 |
| 内存读 | RDx/RAy = M[RAx]; | RDx/RAy = GET\_M(RAx); |
| 内存写 | M[RAx] = RDx/RAy; | SET\_M(RAx, RDx/RAy); |
| 内存读  (RA++或--) | RDx/RAy = M[RAx++]; | RDx/RAy = GET\_M(RAx); RAx+=1\*MMU\_BASE; |
| 内存写  (RA++或--) | M[RAx++] = RDx/RAy; | SET\_M(RAx, RDx/RAy);  RAx+=1\*MMU\_BASE; |
| 临时内存申请 | RSP -= N\*MMU\_BASE; | RSP -= N\*MMU\_BASE; |
| 临时内存释放 | RSP +=N\*MMU\_BASE; | RSP +=N\*MMU\_BASE; |
| 压栈 | push RDx/RAx | push(RDx/RAx) |
| 退栈 | pop RDx/RAx | pop(RDx/RAx) |
| 内存块拷贝 | memcpy/dma指令 | MEMCPY(len, src, dst) |
| 函数声明1 | Sub func;  … Return(n\*MMU\_BASE); n为函数压栈参数个数 | Sub func() {  …  Return(n\*MMU\_BASE);  } |
| 函数声明2 | Sub\_AutoField func;  … Return(n\*MMU\_BASE); n为函数压栈参数个数 | Sub\_AutoField func() {  …  Return\_AutoField(n\*MMU\_BASE);  } |
| 函数调用 | send\_para(RD0);  send\_para(RD1); call func; | 对于Sub类函数使用 call func(); 对于SubAutoField则使用call\_AutoField func(); |

## 四、DSP

本模块提供芯片平台所支持的各种数据处理方法，如：滤波器、FFT、序列运算、ALU等等。还可超前提供规划中的运算模块，比如NLMS加速器、浮点运算器。

在嵌入式系统中，硬件加速器是有效的提高运算效率的手段。如遇到序列运算时应尽量调用本模块的函数。

|  |  |
| --- | --- |
| **ALU** | |
| 清除指定的GRAM块 | Ram\_Clr |
| 给指定的GRAM块设置固定值 | Ram\_Set |
| 双序列加法运算，32bit运算 | Dual\_Ser\_Add32 |
| 双序列减法运算，32bit运算 | Dual\_Ser\_Sub32 |
| 单序列移位运算，配置可选 | Cal\_Single\_Shift |
| 单序列逻辑右移1位运算，32bit运算 | Cal\_Single\_ShiftR1 |
| 双序列加法运算，32bit运算，128DWORD | Add\_DMA\_Wola |
| 双序列加法运算（限幅至16bit），32bit运算 | Add\_LMT |
| 双序列减法运算（限幅至16bit），32bit运算 | Sub\_LMT |
| 单序列加常量，高低16bit都有效 | Cal\_Single\_Add\_Const |
| 32bit序列转16bit序列 | LMT32To16 |
| DMA传输数据 | DMA\_Trans |
| 通过PATH1的DMA传输数据，目标地址为递减模式（倒序） | DMA\_TransDimin\_PATH1 |
| **MAC** | |
| 单序列平方运算 | SingleSerSquare/SingleSerSquareL |
| 双序列乘累加运算 | MultiSum\_Init |
| 单序列乘常量 | MAC\_MultiConst16 |
| 双序列乘常量 | MultiConstH16L16 |
| 双矢量乘法，Q7输出 | MultiVector2 |
| 实部调制序列乘，Q7输出 | ModulationToZero |
| 双序列乘,Q0输出 | DualSequMulti\_Q0 |
| 序列与Const相乘运算 | WienerMultiConst24\_DivQ7\_LMT |
| 双序列乘运算 | WienerSeqMulti\_DivQ7 |
| 实部调制序列乘，Q15输出 | ModulationToZero2 |
| 为单序列乘常量，常量的高低16bit相同 | MAC\_MultiConst16\_Q2207 |
| **FFT** | |
| 64点FFT运算 | FFT\_fix64 |
| 128点FFT运算 | FFT\_Fast128 |
| **FMT(Format序列格式处理模块)** | |
| 提取实部 | Get\_Real |
| 提取虚部 | Get\_Imag |
| 紧凑16bit格式转换为复数格式,虚部置零 | Real\_To\_Complex2 |
| **IIR滤波器** | |
| **滤波器组** | |
| **MATH** | |
| 定点求Power = 10^(RD0/10) | power\_fix |
| 定点求倒数（Q23输出） | recip\_fix |
| 定点求倒数（Q7输出） | recip\_fix\_Q7 |
| 简化版定点求Power = 2^(RD0) | power2\_fix |
| 定点求log2 | log2\_fix |
| **SOC\_Common** | |
| RD0左移RD1次，右补0 | \_Rf\_ShiftL\_Reg |
| RD0右移RD1次，左补0 | \_Rf\_ShiftR\_Reg |
| 无符号乘法 | \_Ru\_Multi |
| 有符号乘法 | \_Rs\_Multi |
| **STA** | |
| 求序列极值 | FindMaxMin |
| 求序列的绝对值累加和 | AbsSum |
| 求序列绝对值的最大值的Index | FindMaxIndex |
| **…** | |

## 五、IO

本模块负责模拟器与电脑文件之间的数据传输操作。比如模拟硬件的ADC取一帧的动作，模拟器里使用getData\_1Frame函数从预置好的某文件中读一帧数据。

## 六、统计信息

本模块用于统计算法实现中的内存资源占用，复杂度估计，时间消耗估计，甚至是硬件平台运行功耗的大致水平。

## 七、图形化调试

本模块用于将过程中数据在一个图形化界面中直观显示，比如：时域曲线、声压级、频谱等。八、函数库

有三种类型的函数调用：

类型A：函数名(参数1,参数2…);

这种方式形如C语言的调用。例如：push(RD0)、pop(RD0)、RD0 = GET\_M(RA0)、寄存器的单目操作等等。

类型B：*call* 函数名;

当函数声明使用Sub关键字时，必须用call调用，必须使用Return(N\*MMU\_BASE)返回。(N为压栈型参数个数，寄存器参数不计入在内)

类型C：*call\_AutoField* 函数名;

当函数声明使用Sub\_AutoField关键字时，必须用call\_AutoField调用，必须使用Return\_AutoField (N\*MMU\_BASE)返回。(N为压栈型参数个数，寄存器参数不计入在内)

另外，类型B和C的传参方式有两种，一是通过寄存器直接传递；二是通过send\_para(x)堆栈传参。

需注意的是：Sub关键字隐含了一次send\_para；Sub\_AutoField关键字则会对RD2,RD3,RA0,RA1进行保护，这种嵌套的深度最多为8层。

### memory.h

push

|  |  |  |
| --- | --- | --- |
| 函数名 | push(x); | 类型A |
| 功能 | 将寄存器压栈，一般与pop组合使用，用于暂存和恢复寄存器值 | |
| 返回值 | 无 | |
| 参数 | x: RD或RA寄存器 | |
| 备注 | 此操作会使得RSP减去4 | |

pop

|  |  |  |
| --- | --- | --- |
| 函数名 | pop(x); | 类型A |
| 功能 | 退栈，将值写入寄存器。一般与push组合使用，用于暂存和恢复寄存器值 | |
| 返回值 | 无 | |
| 参数 | x: RD或RA寄存器 | |
| 备注 | 此操作会使得RSP加上4 | |

GET\_M

|  |  |  |
| --- | --- | --- |
| 函数名 | GET\_M(x); | 类型A |
| 功能 | 读内存，相当于M[RAx] | |
| 返回值 | RA指向的内存值 | |
| 参数 | x: RA寄存器 | |
| 备注 | 暂不支持ROM地址 | |

SET\_M

|  |  |  |
| --- | --- | --- |
| 函数名 | SET\_M(x, y); | 类型A |
| 功能 | 写内存，相当于M[RAx] = RDy; | |
| 返回值 | 无 | |
| 参数 | x: RA寄存器,指向内存某地址  y: RD/RA寄存器 | |
| 备注 | 暂不支持ROM地址 | |

### CData\_io.h

CData\_io类，负责系统的输入输出文件，以及分帧读取和写出。

getData\_1Frame

|  |  |  |
| --- | --- | --- |
| 函数名 | getData\_1Frame  (RAx, frameLen); | 类型A |
| 功能 | 从指定的输入文件中取一帧数据，相当于AD过程 | |
| 返回值 | 实际取到的数据点数，若为0表示异常 | |
| 参数 | RAx: RA寄存器，指向本帧数据要存入的位置  FrameLen: 一帧的数据点数 | |
| 备注 |  | |

outData\_1Frame

|  |  |  |
| --- | --- | --- |
| 函数名 | outData\_1Frame  (RAx, frameLen); | 类型A |
| 功能 | 向指定的输出文件写入一帧数据，相当于DA过程 | |
| 返回值 | 无 | |
| 参数 | RAx: RA寄存器，指向待输出的数据  FrameLen: 一帧的数据点数 | |
| 备注 |  | |

### alu.h

Ram\_Clr

|  |  |  |
| --- | --- | --- |
| 函数名 | Ram\_Clr | 类型C |
| 功能 | 将一段内存置0 | |
| 返回值 | 无 | |
| 参数 | RA1:指定的GRAM块地址(out)  RD1:Dword长度 | |
| 备注 | 无 | |

Ram\_Set

|  |  |  |
| --- | --- | --- |
| 函数名 | Ram\_Set | 类型C |
| 功能 | 给指定的GRAM块设置固定值 | |
| 返回值 | 无 | |
| 参数 | RA1:指定的GRAM块地址(in&out)  RD0:Dword长度  RD1:const值 | |
| 备注 | 无 | |

Dual\_Ser\_Add32

|  |  |  |
| --- | --- | --- |
| 函数名 | Dual\_Ser\_Add32 | 类型C |
| 功能 | 双序列加法运算，32bit运算 | |
| 返回值 | 无 | |
| 参数 | RA0:输入序列1指针，32bit格式序列(out)  RA1:输入序列2指针，32bit格式序列  RD0:序列Dword长度 | |
| 备注 | 无 | |

Dual\_Ser\_Sub32

|  |  |  |
| --- | --- | --- |
| 函数名 | Dual\_Ser\_Sub32 | 类型C |
| 功能 | 双序列减法运算，32bit运算 | |
| 返回值 | 无 | |
| 参数 | RA0:输入序列1指针，32bit格式序列  RA1:输入序列2指针，32bit格式序列  RD1:输出序列指针，32bit格式序列(out)  RD0:序列Dword长度 | |
| 备注 | 无 | |

Cal\_Single\_Shift

|  |  |  |
| --- | --- | --- |
| 函数名 | Cal\_Single\_Shift | 类型C |
| 功能 | 单序列移位运算，配置可选 | |
| 返回值 | 无 | |
| 参数 | RD0:处理位宽+移位处理指令 (例:Op32bit+Rf\_SftR1)详见DMA\_ALU.h  RD1:序列长度  RA0:输入序列指针(out相同) | |
| 备注 | 无 | |

Cal\_Single\_ShiftR1

|  |  |  |
| --- | --- | --- |
| 函数名 | Cal\_Single\_ShiftR1 | 类型C |
| 功能 | 单序列逻辑右移1位运算，32bit运算 | |
| 返回值 | 无 | |
| 参数 | RD0:输入序列指针，32bit格式序列(out)  RD1:输入序列Dword长度 | |
| 备注 | 无 | |

Add\_DMA\_Wola

|  |  |  |
| --- | --- | --- |
| 函数名 | Add\_DMA\_Wola | 类型C |
| 功能 | 双序列加法运算，32bit运算，128DWORD | |
| 返回值 | 无 | |
| 参数 | RA0:输入序列1指针，32bit格式序列(out)  RA1:输入序列2指针，32bit格式序列 | |
| 备注 | 无 | |

Add\_LMT

|  |  |  |
| --- | --- | --- |
| 函数名 | Add\_LMT | 类型C |
| 功能 | 双序列加法运算（限幅至16bit），32bit运算 | |
| 返回值 | 无 | |
| 参数 | RA0:输入序列1指针，32bit格式序列  RA1:输入序列2指针，32bit格式序列  RD1:输出序列指针，32bit格式序列(out)  RD0:序列Dword长度 | |
| 备注 | 无 | |

Sub\_LMT

|  |  |  |
| --- | --- | --- |
| 函数名 | Sub\_LMT | 类型C |
| 功能 | 双序列减法运算（限幅至16bit），32bit运算 | |
| 返回值 | 无 | |
| 参数 | RA0:输入序列1指针，32bit格式序列(out)  RA1:输入序列2指针，32bit格式序列  RD0:序列Dword长度 | |
| 备注 | 无 | |

Cal\_Single\_Add\_Const

|  |  |  |
| --- | --- | --- |
| 函数名 | Cal\_Single\_Add\_Const | 类型C |
| 功能 | 单序列加常量，高低16bit都有效 | |
| 返回值 | 无 | |
| 参数 | RD1:Const  RA0:源地址  RA1:目标地址(out)  RD0:数据长度 | |
| 备注 | 无 | |

LMT32To16

|  |  |  |
| --- | --- | --- |
| 函数名 | LMT32To16 | 类型C |
| 功能 | 32bit序列转16bit序列 | |
| 返回值 | 无 | |
| 参数 | RA0:输入序列指针，32bit格式序列  RA1:输出序列指针，16bit紧凑格式序列(out)  RD0:输出序列Dword长度 | |
| 备注 | 无 | |

DMA\_Trans

|  |  |  |
| --- | --- | --- |
| 函数名 | DMA\_Trans | 类型C |
| 功能 | DMA传输数据 | |
| 返回值 | 无 | |
| 参数 | RA0:源地址  RA1:目标地址(out)  RD0:数据Dword长度 | |
| 备注 | 无 | |

DMA\_TransDimin\_PATH1

|  |  |  |
| --- | --- | --- |
| 函数名 | DMA\_TransDimin\_PATH1 | 类型C |
| 功能 | 通过PATH1的DMA传输数据，目标地址为递减模式（倒序） | |
| 返回值 | 无 | |
| 参数 | RA0:源地址  RA1:目标地址(out)  RD0:数据Dword长度 | |
| 备注 | 无 | |

### FMT\_F.h

Get\_Real

|  |  |  |
| --- | --- | --- |
| 函数名 | DMA\_TransDimin\_PATH1 | 类型C |
| 功能 | 提取实部（系统默认高位为实部）, in, out可以是同一地址 | |
| 返回值 | 无 | |
| 参数 | RA0:输入序列指针，格式[Re | Im]  RA1:输出序列指针，格式[Re(n+1) | Re(n)](out)  RD0:序列长度 | |
| 备注 | 无 | |

Get\_ Imag

|  |  |  |
| --- | --- | --- |
| 函数名 | Get\_Imag | 类型C |
| 功能 | 提取虚部（系统默认低位为虚部）,in,out可以是同一地址 | |
| 返回值 | 无 | |
| 参数 | RA0:输入序列指针，格式[Re | Im]  RA1:输出序列指针，格式[Re(n+1) | Re(n)](out)  RD0:序列长度 | |
| 备注 | 无 | |

Real\_To\_Complex2

|  |  |  |
| --- | --- | --- |
| 函数名 | Real\_To\_Complex2 | 类型C |
| 功能 | 紧凑16bit格式转换为复数格式,虚部置零 | |
| 返回值 | 无 | |
| 参数 | RA0:输入序列指针，格式[Re(n+1) | Re(n)]  RA1:输出序列指针，格式[Re | 0](out)  RD0:序列长度 | |
| 备注 | 无 | |

### mac.h

Multi24

|  |  |  |
| --- | --- | --- |
| 函数名 | Multi24 | 类型A |
| 功能 | 乘法器，通过c语言模拟硬件24位乘法器特性。 | |
| 返回值 | 无 | |
| 参数 | Multi24\_0:乘数，对应MAC硬件内的RA0序列，16bit格式  Multi24\_1:乘数，对应MAC硬件内的RA1序列，16bit格式进来，计算时扩充为24bit计算  RD0:乘法结果，32bit格式 | |
| 备注 | 无 | |

SingleSerSquare

|  |  |  |
| --- | --- | --- |
| 函数名 | SingleSerSquare | 类型C |
| 功能 | 单序列平方运算 | |
| 返回值 | 无 | |
| 参数 | RA0:输入序列指针，紧凑16bit格式，高低位都有数  RA1:输出序列指针，紧凑16bit格式(out)  RA0:Dword长度 | |
| 备注 | 无 | |

MultiSum

|  |  |  |
| --- | --- | --- |
| 函数名 | MultiSum | 类型C |
| 功能 | 单序列平方运算 | |
| 返回值 | RD0:乘累加结果 | |
| 参数 | RA0:输入序列1指针，紧凑16bit格式序列  RA1:输入序列2指针，紧凑16bit格式序列  RD1:输出序列指针，紧凑16bit格式序列(乘累加中间量)  RD0:序列长度 | |
| 备注 | 无 | |

MAC\_MultiConst16

|  |  |  |
| --- | --- | --- |
| 函数名 | MAC\_MultiConst16 | 类型C |
| 功能 | 单序列乘常量 | |
| 返回值 | 无 | |
| 参数 | M[RSP+3\*MMU\_BASE]：X(n) 首地址（字节地址）  M[RSP+2\*MMU\_BASE]：Const 注意要求高16位与低16位相同  M[RSP+1\*MMU\_BASE]：Z(n) 首地址  M[RSP+0\*MMU\_BASE]：数据长度 | |
| 备注 | 无 | |

MultiConstH16L16

|  |  |  |
| --- | --- | --- |
| 函数名 | MultiConstH16L16 | 类型C |
| 功能 | 双序列乘常量,Q15输出 | |
| 返回值 | 无 | |
| 参数 | RA0:输入序列1指针，紧凑16bit格式序列  RA1:输出序列指针，紧凑16bit格式序列(out)  RD1:Const值  RD0:序列长度 | |
| 备注 | 无 | |

MultiVector2

|  |  |  |
| --- | --- | --- |
| 函数名 | MultiVector2 | 类型C |
| 功能 | 双矢量乘法，Q7输出 | |
| 返回值 | 无 | |
| 参数 | RA0:\*in1  RA1:\*in2  RD1:\*out  RD0:序列长度 | |
| 备注 | 无 | |

Mac\_Sim

|  |  |  |
| --- | --- | --- |
| 函数名 | Mac\_Sim | 类型A |
| 功能 | 模拟Mac硬件数据路径 | |
| 返回值 | 无 | |
| 参数 | addr\_0:data0地址  addr\_1:data1地址  addr\_out:out地址  Const\_Reg：常数值  Config\_Reg：配置参数，详情见MAC数据路径示意图  len:序列长度 | |
| 备注 | 本函数用来模拟MAC流程，并不全面。如需使用乘累加器，请使用MultiSum函数。 | |

Multi24\_16x24

|  |  |  |
| --- | --- | --- |
| 函数名 | Multi24\_16x24 | 类型A |
| 功能 | 乘法器，通过c语言模拟硬件24位乘法器特性。  主要被Mac\_Sim函数调用。 | |
| 返回值 | 无 | |
| 参数 | Multi24\_0:乘数，对应MAC硬件内的RA0序列，16bit格式  Multi24\_1:乘数，对应MAC硬件内的RA1序列，24bit格式  RD0:乘法结果，32bit格式 | |
| 备注 | 无 | |

ModulationToZero

|  |  |  |
| --- | --- | --- |
| 函数名 | ModulationToZero | 类型C |
| 功能 | 实部调制序列乘Q7 | |
| 返回值 | 无 | |
| 参数 | RA0: x(n), [H16|L16]格式的32位操作数序列;  RA1: y(n), [H16|0]格式的高16位操作数序列;  RD1: z(n) = x(n).\*y(n)的目标地址;  RD0: 数据长度 | |
| 备注 | 无 | |

DualSequMulti\_Q0

|  |  |  |
| --- | --- | --- |
| 函数名 | DualSequMulti\_Q0 | 类型C |
| 功能 | 双序列乘,Q0输出 | |
| 返回值 | 无 | |
| 参数 | RA0:输入序列1指针,紧凑16bit格式  RA1:输入序列2指针,紧凑16bit格式  RD0:Dword长度 | |
| 备注 | 无 | |

DualSequMulti\_Q0

|  |  |  |
| --- | --- | --- |
| 函数名 | DualSequMulti\_Q0 | 类型C |
| 功能 | 双序列乘,Q0输出 | |
| 返回值 | RD1:输出序列指针,紧凑16bit格式(out) | |
| 参数 | RA0:输入序列1指针,紧凑16bit格式  RA1:输入序列2指针,紧凑16bit格式  RD0:Dword长度 | |
| 备注 | 实虚部平方只能采用Q0方式输出,不可以Q1带限幅输出; | |

WienerMultiConst24\_DivQ7\_LMT

|  |  |  |
| --- | --- | --- |
| 函数名 | WienerMultiConst24\_DivQ7\_LMT | 类型C |
| 功能 | 序列与Const相乘运算 | |
| 返回值 | 无 | |
| 参数 | RA0:输入序列1指针,高16bit格式序列  RA1:输出序列指针,结果为: QM1L[22:7],QM0L[22:7]  RD1:Const值,低8位为0  RD0:序列长度 | |
| 备注 | 无 | |

WienerSeqMulti\_DivQ7

|  |  |  |
| --- | --- | --- |
| 函数名 | WienerSeqMulti\_DivQ7 | 类型C |
| 功能 | 双序列乘运算 | |
| 返回值 | RD1:输出序列指针,结果为；QM1L[22:7],QM0L[22:7] | |
| 参数 | RA0:输入序列1指针,高16bit格式序列  RA1:输入序列2指针,高16bit格式序列  RD0:序列长度 | |
| 备注 | 无 | |

ModulationToZero2

|  |  |  |
| --- | --- | --- |
| 函数名 | ModulationToZero2 | 类型C |
| 功能 | 实部调制序列乘Q15 | |
| 返回值 | 无 | |
| 参数 | RA0:表地址  RA1:操作数地址  RD1:目标地址  RD0:数据长度对应的TimerNum值(Dword长度\*3+3) | |
| 备注 | 无 | |

MAC\_MultiConst16\_Q2207

|  |  |  |
| --- | --- | --- |
| 函数名 | MAC\_MultiConst16\_Q2207 | 类型C |
| 功能 | 为单序列乘常量操作配置DMA\_Ctrl参数 | |
| 返回值 | 无 | |
| 参数 | M[RSP+3\*MMU\_BASE]：X(n) 首地址（字节地址）  M[RSP+2\*MMU\_BASE]：Const 注意要求高16位与低16位相同  M[RSP+1\*MMU\_BASE]：Z(n) 首地址  M[RSP+0\*MMU\_BASE]：数据长度 | |
| 备注 | 无 | |

### Math\_F.h

power\_fix

|  |  |  |
| --- | --- | --- |
| 函数名 | power\_fix | 类型C |
| 功能 | 定点求Power = 10^(RD0/10) | |
| 返回值 | RD0: 计算结果的Q8值 | |
| 参数 | RD0: 输入数据的Q0值 | |
| 备注 | 以查表法实现 | |

recip\_fix

|  |  |  |
| --- | --- | --- |
| 函数名 | recip\_fix | 类型C |
| 功能 | 定点求倒数 | |
| 返回值 | RD0: 计算结果的Q23值 | |
| 参数 | RD0: 输入数据的Q0值 | |
| 备注 | 以查表法实现 | |

recip\_fix\_Q7

|  |  |  |
| --- | --- | --- |
| 函数名 | recip\_fix | 类型C |
| 功能 | 定点求倒数 | |
| 返回值 | RD0: 计算结果的Q7值 | |
| 参数 | RD0: 输入数据的Q7值 | |
| 备注 | 以查表法实现 | |

power2\_fix

|  |  |  |
| --- | --- | --- |
| 函数名 | power2\_fix | 类型C |
| 功能 | 定点求Power = 2^(RD0) | |
| 返回值 | RD0: 计算结果的Q8值 | |
| 参数 | RD0: 输入数据的Q8值 | |
| 备注 | 以查表法实现 | |

log2\_fix

|  |  |  |
| --- | --- | --- |
| 函数名 | log2\_fix | 类型C |
| 功能 | 计算以2为底的对数值 | |
| 返回值 | RD0: 计算结果的Q15值 | |
| 参数 | RD0: 输入数据的Q0值 | |
| 备注 | 以查表法实现 | |

### SOC\_Common\_F.h

\_Rf\_ShiftL\_Reg

|  |  |  |
| --- | --- | --- |
| 函数名 | \_Rf\_ShiftL\_Reg | 类型C |
| 功能 | RD0左移RD1次，右补0 | |
| 返回值 | RD0: 移位结果 | |
| 参数 | RD0：被移动的数  RD1：要移的次数 | |
| 备注 | 无 | |

\_Rf\_ShiftR\_Reg

|  |  |  |
| --- | --- | --- |
| 函数名 | \_Rf\_ShiftL\_Reg | 类型C |
| 功能 | RD0右移RD1次，左补0 | |
| 返回值 | RD0: 移位结果 | |
| 参数 | RD0：被移动的数  RD1：要移的次数 | |
| 备注 | 无 | |

\_Ru\_Multi

|  |  |  |
| --- | --- | --- |
| 函数名 | \_Ru\_Multi | 类型C |
| 功能 | 计算[zh,zl]=x\*y; x,y,zh,zl均为32比特的整数,zh,zl分别为结果的高部.低部; | |
| 返回值 | RD0:zl-乘积低部;  RD1:zh-乘积高部; | |
| 参数 | RD0:乘数x;  RD1:乘数y; | |
| 备注 | 无 | |

\_Rs\_Multi

|  |  |  |
| --- | --- | --- |
| 函数名 | \_Rs\_Multi | 类型C |
| 功能 | 计算[zh,zl]=x\*y; x,y,zh,zl均为32比特的整数,zh,zl分别为结果的高部.低部; | |
| 返回值 | RD0:zl-乘积低部;  RD1:zh-乘积高部; | |
| 参数 | RD0:乘数x;  RD1:乘数y; | |
| 备注 | 无 | |

### STA\_F.h

FindMaxMin

|  |  |  |
| --- | --- | --- |
| 函数名 | FindMaxMin | 类型C |
| 功能 | 求序列极值 | |
| 返回值 | RD0:最大值  RD1:最小值 | |
| 参数 | RD0:数据地址  RD1:数据长度 | |
| 备注 | 无 | |

AbsSum

|  |  |  |
| --- | --- | --- |
| 函数名 | AbsSum | 类型C |
| 功能 | 求序列的绝对值累加和 | |
| 返回值 | RD0:绝对值累加和 | |
| 参数 | RD0:数据地址  RD1:数据长度 | |
| 备注 | 无 | |

FindMaxIndex

|  |  |  |
| --- | --- | --- |
| 函数名 | FindMaxIndex | 类型C |
| 功能 | 求序列绝对值的最大值的Index（STA2）,index以DWORD为单位,0..L-1 | |
| 返回值 | RD0:最大值的Index | |
| 参数 | RD0:数据地址  RD1:数据长度 | |
| 备注 | 无 | |