



# CK 610 用户手册

## USERGUIDE

□

**C-Sky Confidential**

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).



## 声明：

杭州中天微系统有限公司(C-SKY Microsystems Co.,Ltd)保留本文档的所有权利。  
本文档的内容有可能发生更改、更新、删除、变动，恕不另行通知。

版权所有 © 2007-2017 杭州中天微系统有限公司

公司地址：杭州市西湖区西斗门路 3 号天堂软件园 A 座 15 层

邮政编码：310012

电话：0571-88157059

传真：0571-88157059-8888

主页：[www.c-sky.com](http://www.c-sky.com)

E-mail：[info@c-sky.com](mailto:info@c-sky.com)



**C-Sky Confidential**

The information contained herein is confidential and proprietary and is not to be disclosed outside of  
Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

## 目录：

E-MAIL: INFO@C-SKY.COM.....	1
1. 简介.....	11
1.1. 特点.....	12
1.2. 微体系结构.....	12
1.3. 编程模型.....	13
1.4. 数据格式.....	14
1.5. 操作数寻址能力.....	15
1.6. 指令集一览.....	15
2. 命名规则.....	19
2.1. 符号.....	19
2.2. 术语.....	19
3. 寄存器描述.....	21
3.1. 普通用户编程模式.....	21
3.1.1. 通用寄存器.....	21
3.1.2. 程序计数器.....	21
3.1.3. 条件码／进位标志位.....	21
3.2. 超级用户编程模式.....	22
3.2.1. 可选择寄存器.....	22
3.2.2. 处理器状态寄存器 (PSR, CR0) .....	23
3.2.3. 向量基址寄存器 (VBR, CR1) .....	25
3.2.4. 异常保留寄存器 (CR2~CR5) .....	25
3.2.5. 存储寄存器 SS0~SS4 (CR6~CR10) .....	25
3.2.6. 全局控制寄存器 (GCR, CR11) .....	26
3.2.7. 全局状态寄存器 (GSR, CR12) .....	26
3.2.8. 产品序号寄存器 (CPUIDRR, CR13) .....	26
3.2.9. DSP 控制状态寄存器 <sup>610E</sup> (DCSR, CR14) .....	27
3.2.10. 协处理器窗口寄存器 <sup>CP</sup> (CPWR, CR15) .....	27
3.2.11. 高速缓存功能设置寄存器 (CFR, CR17) .....	28
3.2.12. 高速缓存配置寄存器 (CCR, CR18) .....	29
3.2.13. 可高缓和访问权限配置寄存器 (CAPR, CR19) .....	30
3.2.14. 保护区控制寄存器 (PACR, CR20) .....	30
3.2.15. 保护区选择寄存器 (PRSR, CR21) .....	31
3.2.16. 高速缓存索引寄存器 (CIR, CR22) .....	31
3.2.17. CPU HINT 寄存器 (CHR, CR<30,0>) .....	31
3.2.18. MGU 使用操作.....	32
3.2.19. MMU 使用操作.....	32
3.2.20. MMU 索引寄存器 <sup>610M</sup> (MIR, CP15_CR0).....	33
3.2.21. MMU 随机寄存器 <sup>610M</sup> (MRR, CP15_CR1).....	34
3.2.22. MMU EntryLo0 和 EntryLo1 寄存器 <sup>610M</sup> (MEL0 & MEL1, CP15_CR2 & CP15_CR3).....	34
3.2.23. MMU EntryHi/Bad VPN 寄存器 <sup>610M</sup> (MEH, CP15_CR4).....	35
3.2.24. MMU 上下文寄存器 <sup>610M</sup> (MCR, CP15_CR5).....	35
3.2.25. MMU 页掩码寄存器 <sup>610M</sup> (MPR, CP15_CR6).....	35

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

3.2.26. MMU 圈连寄存器 <sup>610M</sup> (MWR, CP15_CR7).....	36
3.2.27. MMU 控制指令寄存器 <sup>610M</sup> (MCIR, CP15_CR8).....	36
3.2.28. MMU PGD 基址寄存器(MPGD, CP15_CR29).....	37
3.2.29. MMU SSEG0 配置寄存器(MSA0, CP15_CR30).....	37
3.2.30. MMU SSEG1 配置寄存器(MSA1, CP15_CR31).....	37
3.2.31. jTLB 表项结构 <sup>610M</sup> .....	38
3.2.32. 高速暂存器 (SPM) .....	38
3.2.33. 指令 SPM 寄存器(CP15_CR9).....	39
3.2.34. 数据 SPM 寄存器(CP15_CR10).....	40
3.2.35. DMA 使能寄存器(CP15_CR11).....	41
3.2.36. DMA 控制寄存器(CP15_CR12).....	41
3.2.37. DMA 状态寄存器(CP15_CR13).....	42
3.2.38. DMA 外部起始地址寄存器(CP15_CR14).....	42
3.2.39. DMA 内部起始地址寄存器(CP15_CR15).....	43
3.2.40. DMA 传输大小寄存器(CP15_CR16).....	43
<b>4. 指令集.....</b>	<b>44</b>
4.1. 指令类型和寻址方式.....	44
4.1.1. 寄存器操作指令.....	44
4.1.2. 内存访问指令.....	45
4.1.3. 跳转指令.....	46
4.2. 指令码表.....	47
<b>5. 指令流水线.....</b>	<b>53</b>
<b>6. 异常处理.....</b>	<b>57</b>
6.1. 异常处理概述.....	57
6.2. 异常类型.....	58
6.2.1. 重启异常 (向量偏移 0X0) .....	59
6.2.2. 未对齐访问异常 (向量偏移 0X4) .....	59
6.2.3. 访问错误异常 (向量偏移 0X8) .....	59
6.2.4. 除以零异常 (向量偏移 0X0C) .....	60
6.2.5. 非法指令异常 (向量偏移 0X10) .....	60
6.2.6. 特权违反异常 (向量偏移 0X14) .....	60
6.2.7. 跟踪异常 (向量偏移 0X18) .....	60
6.2.8. 断点异常 (向量偏移 0X1C) .....	60
6.2.9. 不可恢复错误异常 (向量偏移 0X20) .....	61
6.2.10. IDLY4 异常 (异常偏移 0X24) .....	61
6.2.11. 中断异常.....	61
6.2.12. TLB 不可恢复异常 <sup>610M</sup> (向量偏移 0X34) .....	62
6.2.13. TLB 失配异常 <sup>610M</sup> (向量偏移 0X38) .....	62
6.2.14. TLB 修改异常 <sup>610M</sup> (向量偏移 0X3C) .....	62
6.2.15. 陷阱指令异常 (向量偏移 0X40 – 0X4C) .....	63
6.2.16. TLB 读无效异常 <sup>610M</sup> (向量偏移 0X50) .....	63
6.2.17. TLB 写无效异常 <sup>610M</sup> (向量偏移 0X54) .....	63
6.2.18. 协处理器中断和异常 <sup>CP</sup> (向量偏移 0X78) .....	63
6.3. 异常优先级.....	63
6.3.1. 发生待处理的异常时调试请求.....	64
6.4. 异常返回.....	64

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

7. 接口信号.....	65
7.1. 时钟信号.....	65
7.2. 接口信号.....	65
7.2.1. 信号类型.....	65
7.2.2. 信号功能.....	66
7.3. 内部时钟与外部时钟的对齐.....	73
8. 接口总线协议.....	76
8.1. 外部接口信号列表.....	76
8.2. 系统总线传输协议.....	77
8.2.1. 概览.....	77
8.2.2. C-SKY CPU burst 传输.....	77
8.2.3. 一些典型的传输模式.....	77
8.2.4. 总线异常.....	83
9. 工作模式转换.....	84
9.1. C-SKY CPU 工作模式及其转换.....	84
9.1.1. 正常工作模式.....	84
9.1.2. 低功耗模式.....	84
9.1.3. 调试模式.....	85
附录 A MGU 设置示例.....	86
附录 B MMU 设置示例.....	88
附录 B 指令术语表.....	90
ABS——绝对值指令.....	91
ADDC——无符号带进位 C 加法指令.....	92
ADDI——无符号立即数加法指令.....	93
ADDU——无符号加法指令.....	94
AND——按位与指令.....	95
ANDI——立即数按位与指令.....	96
ANDN——按位与非指令.....	97
ASR——算术右移指令（动态）.....	98
ASRC——算术右移一位到 C 进位位.....	99
ASRI——立即数算术右移（静态）.....	100
BCLRI——立即数位清零.....	101
BF——C 为 0 条件跳转指令.....	102
BGENI——立即数位产生指令（静态）.....	103
BGENR——寄存器位产生指令.....	104
BKPT——断点指令.....	105
BMASKI——立即数位屏蔽产生指令.....	106
BR——无条件跳转指令.....	107
BREV——位倒序指令.....	108
BSETI——立即数置位指令.....	109
BSR——跳转到子程序指令.....	110
BT——C 为 1 条件跳转指令.....	111
BTSTI——立即数位测试指令.....	112
CLRF——C 为 0 清零指令.....	113

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of  
Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

CLRT——C 为 1 清零指令.....	114
CMPHS——大于等于比较指令.....	115
CMPLT——小于比较指令.....	116
CMPLTI——立即数小于比较指令.....	117
CMPNE——不等比较指令.....	118
CMPNEI——立即数不等比较指令.....	119
DECf——C 为 0 条件减 1 指令.....	120
DECGT——减 1 大于置位指令.....	121
DECLT——减 1 小于置位指令.....	122
DECNE——减 1 不等置位指令.....	123
DECT——C 为 1 条件减 1 指令.....	124
DIVS——有符号寄存器除法指令.....	125
DIVU——无符号寄存器除法指令.....	126
DOZE——进入低功耗睡眠模式指令.....	127
FF1——快速找 1 指令.....	128
IDLY4——中断识别禁止指令.....	129
INCF——C 为 0 寄存器增 1 指令.....	130
INCT——C 为 1 寄存器增 1 指令.....	131
IXH——索引半字指令.....	132
IXW——索引字指令.....	133
JMP——无条件转移指令.....	134
JMPI——无条件间接转移指令.....	135
JSR——无条件子程序转移指令.....	136
JSRI——无条件子程序间接转移指令.....	137
LD.[BHW]——从存储器装入寄存器.....	138
LDM——从存储器装入多个寄存器.....	139
LDQ——从存储器装入 4 个寄存器.....	140
LRW——从存储器读入.....	141
LSLC——逻辑左移 1 位指令.....	143
LSLI——立即数逻辑左移指令（静态）.....	144
LSR——逻辑右移指令（动态）.....	145
LSRC——逻辑右移 1 位指令.....	146
LSRI——立即数逻辑右移指令（静态）.....	147
MAC <sup>620</sup> ——乘法累加指令.....	148
MFCR——控制寄存器数据读传送指令.....	149
MFHI <sup>620</sup> ——高位寄存器数据读传送指令.....	150
MFHIS <sup>610E</sup> ——HI 寄存器数据饱和值传送指令.....	151
MFLO <sup>620</sup> ——低位寄存器数据读传送指令.....	152
MFLOS <sup>610E</sup> ——Lo 寄存器数据饱和值传送指令.....	153
MOV——逻辑数据传送指令.....	154
MOVF——C 为 0 逻辑数据传送指令.....	155
MOVI——立即数逻辑数据传送指令.....	156
MOVT——C 为 1 逻辑数据传送指令.....	157
MTCR——控制寄存器数据写传送指令.....	158
MTHI <sup>620</sup> ——高位寄存器数据写传送指令.....	159
MTLO <sup>620</sup> ——低位寄存器数据写传送指令.....	160
MULS <sup>610E</sup> ——有符号 32 位乘法指令.....	161
MULSA <sup>610E</sup> ——有符号 32 位乘累加指令.....	162

**C-Sky Confidential**

The information contained herein is confidential and proprietary and is not to be disclosed outside of  
Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

<b>MULSH</b> ——16 位有符号乘法指令	163
<b>MULSHA<sup>610E</sup></b> ——有符号 16 位乘累加指令	164
<b>MULSHS<sup>610E</sup></b> ——有符号 16 位乘累减指令	165
<b>MULSS<sup>610E</sup></b> ——有符号 32 位乘累减指令	166
<b>MULSW<sup>610E</sup></b> ——有符号 16x32 乘法指令	167
<b>MULSWA<sup>610E</sup></b> ——有符号 16x32 乘累加指令	168
<b>MULSWS<sup>610E</sup></b> ——有符号 16x32 乘累减指令	169
<b>MULT</b> ——乘法指令	170
<b>MULU<sup>610E</sup></b> ——无符号 32 位乘法指令	171
<b>MULUA<sup>610E</sup></b> ——无符号 32 位乘累加	172
<b>MULUS<sup>610E</sup></b> ——无符号 32 位乘累减指令	173
<b>MVC</b> ——位 C 传送指令	174
<b>MVCV</b> ——位 C 取反传送指令	175
<b>MVTC<sup>610E</sup></b> ——溢出位复制到 C 位指令	176
<b>NOT</b> ——按位非指令	177
<b>OMFLIP<sup>620</sup></b> ——基于 OMEGA-FLIP 网络的总体置换指令	178
<b>OR</b> ——按位或指令	179
<b>PSRCLR</b> ——PSR 清零指令	180
<b>PSRSET</b> ——PSR 置位指令	181
<b>RFI</b> ——快速中断返回指令	182
<b>ROTLI</b> ——立即数循环左移指令	183
<b>RSUB</b> ——减法指令	184
<b>RSUBI</b> ——立即数减法指令	185
<b>RTE</b> ——异常情况返回指令	186
<b>SEXTB</b> ——字节符号扩展指令	187
<b>SEXTH</b> ——半字符号扩展指令	188
<b>ST.[B,H,W]</b> ——从寄存器存入存储器	189
<b>STM</b> ——多个寄存器存储到存储器指令	190
<b>STOP</b> ——进入低功耗暂停模式指令	191
<b>STQ</b> ——4 个寄存器存入存储器指令	192
<b>SUBC</b> ——无符号带进位减法指令	193
<b>SUBI</b> ——无符号立即数减法指令	194
<b>SUBU</b> ——无符号减法指令	195
<b>SYNC</b> ——CPU 同步指令	196
<b>TRAP</b> ——无条件操作系统陷阱指令	197
<b>TST</b> ——零测试指令	198
<b>TSTNBZ</b> ——无字节等于零寄存器测试指令	199
<b>VMULSH<sup>610E</sup></b> ——两路有符号 16 位乘法指令	200
<b>VMULSHA<sup>610E</sup></b> ——两路有符号 16 位乘累加指令	201
<b>VMULSHS<sup>610E</sup></b> ——两路有符号 16 位乘累减指令	202
<b>VMULSW<sup>610E</sup></b> ——两路有符号 16x32 乘法指令	203
<b>VMULSWA<sup>610E</sup></b> ——两路有符号 16x32 乘累加指令	204
<b>VMULSWS<sup>610E</sup></b> ——两路有符号 16x32 乘累减指令	205
<b>WAIT</b> ——停止执行等待中断指令	206
<b>XOR</b> ——按位异或指令	207
<b>XSR</b> ——扩展右移指令	208
<b>XTRB0</b> ——提取字节 0 到 R1 并进行零扩展指令	209
<b>XTRB1</b> ——提取字节 1 到 R1 并进行零扩展指令	210

**C-Sky Confidential**

The information contained herein is confidential and proprietary and is not to be disclosed outside of  
Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

XTRB2——提取字节 2 到 R1 并进行零扩展指令.....	211
XTRB3——提取字节 3 到 R1 并进行零扩展指令.....	212
ZEXTB——字节零扩展指令.....	213
ZEXTH——半字零扩展指令.....	214



**C-Sky Confidential**

The information contained herein is confidential and proprietary and is not to be disclosed outside of  
Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

## 图表目录：

图表 1-1 C-SKY CK610 系列各种类型处理器的特点.....	11
图表 1-2 编程模型.....	13
图表 1-3 内存中的数据组织形式.....	15
图表 1-4 寄存器中的数据组织结构.....	15
图表 1-5 C-SKY CPU 的指令集.....	15
图表 3-1 普通用户编程模式寄存器.....	21
图表 3-2 超级用户编程模式附加资源.....	22
图表 3-3 处理器状态寄存器.....	23
图表 3-4 TM[1:0]编码与相对应的工作模式.....	24
图表 3-5 基址向量寄存器.....	25
图表 3-6 产品序号寄存器.....	26
图表 3-7 DSP 控制状态寄存器.....	27
图表 3-8 协处理器窗口寄存器.....	27
图表 3-9 高速缓存功能设置寄存器.....	28
图表 3-10 高速缓存配置寄存器.....	29
图表 3-11 CK610, CK620 内存保护设置.....	29
图表 3-12 CK610M 内存保护设置.....	30
图表 3-13 可高缓和访问权限配置寄存器.....	30
图表 3-14 访问权限设置.....	30
图表 3-15 保护区控制寄存器.....	30
图表 3-16 保护区大小配置和其对基址要求.....	31
图表 3-17 保护区选择寄存器.....	31
图表 3-18 MMU 索引寄存器.....	33
图表 3-19 MMU ENTRYLO0 和 ENTRYLO1 寄存器.....	34
图表 3-20 MMU ENTRYHi/BAD VPN 寄存器.....	35
图表 3-21 MMU 上下文寄存器.....	35
图表 3-22 MMU 页掩码寄存器.....	36
图表 3-23 页掩码编码.....	36
图表 3-24 MMU 控制指令寄存器.....	36
图表 3-25 MMU PGD 基址寄存器.....	37
图表 3-26 MMU ENTRYLO0 和 ENTRYLO1 寄存器.....	37
图表 3-27 MMU ENTRYLO0 和 ENTRYLO1 寄存器.....	38
图表 3-28 JTLB 表项.....	38
图表 3-29 压缩的页掩码编码.....	38
图表 3-30 指令 SPM 寄存器.....	39
图表 3-31 数据 SPM 寄存器.....	40
图表 3-32 DMA 使能寄存器.....	41
图表 3-33 DMA 控制寄存器.....	41
图表 3-34 DMA 状态寄存器.....	42
图表 3-35 DMA 外部起始地址寄存器.....	43
图表 3-36 DMA 内部起始地址寄存器.....	43
图表 3-37 DMA 传输大小寄存器.....	43
图表 4-1 一元寄存器寻址格式.....	44
图表 4-2 二元寄存器寻址格式.....	44
图表 4-3 寄存器五位立即数寻址格式.....	45
图表 4-4 寄存器五位立即偏移量寻址格式.....	45

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

图表 4-5 寄存器七位立即数寻址格式.....	45
图表 4-6 寄存器七位立即数寻址格式.....	45
图表 4-7 倍乘四位立即数寻址格式.....	45
图表 4-8 加载/存储寄存器象限格式.....	46
图表 4-9 多寄存器载入/存取格式.....	46
图表 4-10 加载相对字格式.....	46
图表 4-11 11 比特移位偏移寻址方式.....	46
图表 4-12 寄存器寻址方式.....	46
图表 4-13 间接寻址指令格式.....	47
图表 4-14 指令编码表.....	47
图表 4-15 指令编码表参数说明.....	52
图表 5-1 各级流水线作用.....	53
图表 5-2 指令执行时的数据流向.....	54
图表 5-3 单周期指令流水线重叠执行.....	54
图表 5-4 LD/ST 指令执行过程.....	54
图表 5-5 乘法指令执行过程.....	54
图表 5-6 除法指令执行过程.....	54
图表 5-7 BR, BSR 指令的跳转和 BT, BF 指令预测跳转且预测正确时执行过程.....	55
图表 5-8 BT, BF 指令预测不跳转且预测正确时执行过程.....	55
图表 5-9 JMP, BT 和 BF 指令预测不正确时的执行过程.....	55
图表 5-10 JSRI 指令的执行过程.....	55
图表 5-11 简单的指令流水线执行过程.....	56
图表 5-12 带有等待状态的指令流水执行过程.....	56
图表 5-13 LRW 指令与访问存储指令混合的执行过程.....	56
图表 6-1 异常向量分配.....	58
图表 6-2 中断接口信号.....	61
图表 6-3 中断处理过程.....	62
图表 6-4 异常优先级.....	63
图表 7-1 C-SKY CPU 的时钟管理方式.....	65
图表 7-2 C-SKY CPU 接口信号命名规则.....	66
图表 7-3 C-SKY CPU 接口信号.....	66
图表 7-4 C-SKY CPU 各个信号的功能、类型、定义.....	66
图表 7-5CPU_CLK 和 SYS_CLK 1:1 波形图.....	73
图表 7-6 CPU_CLK 和 SYS_CLK 2:1 波形图.....	73
图表 7-7 CPU_CLK 和 SYS_CLK 3:1 波形图.....	73
图表 7-8 CPU_CLK 和 SYS_CLK 4:1 波形图.....	74
图表 7-9 CPU_CLK 和 SYS_CLK 5:1 波形图.....	74
图表 7-10 CPU_CLK 和 SYS_CLK 6:1 波形图.....	74
图表 7-11 CPU_CLK 和 SYS_CLK 7:1 波形图.....	74
图表 7-12 CPU_CLK 和 SYS_CLK 8:1 波形图.....	75
图表 8-1 外部接口信号描述列表.....	76
图表 8-2 简单传输.....	78
图表 8-3 有等待状态的传输.....	78
图表 8-4 多重传输.....	79
图表 8-5 传输类型编码.....	79
图表 8-6 传输类型实例.....	80
图表 8-7 HRESP[1:0]响应类型描述.....	80
图表 8-8 具有重传的传输.....	81

## C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of  
Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

图表 8-9 错误响应.....	81
图表 8-10 增量突发传输.....	82
图表 8-11 四拍回绕突发传输.....	83
图表 8-12 总线异常控制循环.....	83
图表 9-1 CPU 的各种工作状态示意图.....	84



**C-Sky Confidential**

The information contained herein is confidential and proprietary and is not to be disclosed outside of  
Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

## 1. 简介

C-SKY CPU 是面向嵌入式系统和 SoC 应用领域的 32 位高性能低功耗嵌入式 CPU 核。C-SKY CPU 采用自主设计的体系结构和微体系结构并具有可扩展指令、可配置硬件资源、可重新综合、易于集成等优点，其在功耗和电源管理上的表现也很出色，可以通过静态设计、动态电源管理和低电压供电来减少功耗，也可以通过进入省电模式来节省功耗，还可以实时地关断内部功能模块。C-SKY CPU CK610 系列处理器在原有处理器的基础上，充分考虑了处理器的性能，功耗和面积等因素，采用双发射指令，乱序执行的机制，自主研发而成的新一代处理器。

目前，杭州中天微系统有限公司拥有的 C-SKY CK610 系列处理器 IP 核包括 CK610、CK620、CK610E、CK610S、CK610M、CK610F 等多种类型。这一系列 IP 核是在保留了 CK610 高性能低功耗的优点的基础上，为适应不同应用背景，所设计的一系列第二代 C-SKY CPU 处理器 IP 核。图表 简介-1 简单介绍了 C-SKY CPU CK610 系列处理器的各自特点。同时由于各种处理器在设计时考虑了正交性（除 CK620 和 CK610E 外），所以用户可以根据自身应用的特点，整合多种处理器的特点。如：CK610EMF 包含了 CK610E、CK610M 和 CK610F 三种类型处理器的特点。

图表 简介-1 C-SKY CK610 系列各种类型处理器的特点

处理器名称	处理器特点
CK610	支持原有 CK510 指令集，高性能，低功耗处理器
CK620	在 CK610 的基础上，增加了对 OMFLIP、MAC、MTLO、MTHI、MFLO 和 MFHI 指令的支持
CK610E	在 CK610 的基础上，增加了对 MFHIS、MFLOS、MULSHA、MULSHS、MULSWA、MULSWS、MULS、MULSA、MULSS、MULU、MULUA、MULUS、MVTC、VMULSH、VMULSHA、VMULSHS、VMULSW、VMULSWA 和 VMULSWS 多种 DSP 指令的支持
CK610S	在 CK610 的基础上，增加了数据 SPM 和指令 SPM，可以在软件工作者配置的情况下，使得常用指令和数据常驻高速缓存（SPM）
CK610M	在 CK610 的基础上，增加了内存管理单元 MMU
CK610F	在 CK610 的基础上，增加了浮点协处理器单元

### 1.1. 特点

CK610 处理器的主要特点如下：

**C-Sky Confidential**

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

- 高性能的 32 位数据长度, 16 位指令长度;
- 0.13um TSMC 工艺, 280MHz 工作频率 (最恶劣情况), 420MHz 工作频率 (正常情况), 392 dhystone MIPS;
- 精简指令集计算机结构 (RISC) ;
- 8 级流水线, 完全对软件透明;
- 双发射指令, 乱序执行;
- 设计功耗 0.4mW/MHz;
- CK610 典型面积 (TSMC, 8K 指令 cache 和 8K 数据 cache) : 0.13um 工艺, 2.25 mm<sup>2</sup>;
- 支持 AMBA2.0/AMBA3.0 内部总线与接口;
- 两级跳转预测;
- 支持返回地址栈;
- 高性能片上高速缓存;
- 支持 big endian 和 little endian;
- 支持直写和写回操作的数据 cache;
- 内部硬件调试模块支持片上硬件调试;
- 支持快速中断, 支持向量中断和自动向量中断;
- 指令高缓和数据高缓大小可变;
- 支持指令重构。

## 1.2. 微体系统结构

C-SKY CPU CK610 系列使用了 8 级流水线结构。

内存管理单元可以让超级用户自由定义 4 个地址空间的访问权限, 权限划分为: 不可读写/只读/可读写。4 个地址空间可重叠, 从而可形成各种访问控制模式。

CK610M 包含片内 MMU, 其具有 4 表项全相联的数据 μTLB、2 表项全相联的指令 μTLB 和 32 表项 2 路组相联 jTLB。提供 2 级 TLB 匹配: μTLB 提升转换速度, 对用户透明; jTLB 提高匹配率, 用户可配置。如果在 μTLB 中没有 TLB 失配, 1 个时钟周期产生物理地址; 如果在 μTLB 中存在 TLB 失配, 但在 jTLB 中没有 TLB 失配, 3 个时钟周期产生物理地址; 如果在 jTLB 中存在 TLB 失配, 软件支持再填充 jTLB。

指令提取单元, 可以配备高速缓存, 采用关键指令先取和发射以及添加指令暂存器等手段来增加指令的发射效率, 利用先进二级指令跳转预测、全局和局部结合的记录来减少因为转移指令造成的流水线空闲, 预测准确率高达 94%。

存储单元支持不间断流水线, 开辟了八个写缓冲区, 具有内部 Bypass 机制, 利用快速退休加快指令的执行速度。存取指令可以操作字节、半字和字, 并可以在字节和半字操作时自动扩展 0。这些存取指令可以流水执行, 使得数据吞吐量达到一个周期存取一个数据。

总线接口单元支持突发传送, 支持关键字优先的地址访问, 可以在不同的系统时钟与 CPU 时钟比例 (1:1, 1:2, 1:3, 1:4, 1:5, 1:6, 1:7, 1:8) 下工作。

标准 JTAG 调试接口支持各种调试方式, 包括软件设置断点方式、硬件设置断点方式、单步和多步指令跟踪、跳转指令跟踪等 7 种方式, 可以在线调试 CPU、通用寄存器 (GPR)、可选择寄存器 (AGPR)、协处理器 0 (CP0) 和内存。

指令执行单元利用指令依赖表格和操作数前馈来有效地处理数据竞争, 实现高性能的指令发射。高性能的指令退休通过 8 个退休缓存器组的非序回收, 并行回收, 按序退休, 快速退休来实现。支持原子操作, 自动分解 LDM/STM 指令。

算术运算单元包括两个 32 位的算术逻辑单元 (ALU), 两个桶型移位器和一个流水线乘法器。算术逻辑运算绝大部分可以在一个周期内完成。

16 个通用寄存器用于提供源操作数和存放指令执行结果。寄存器 R0 通常用作当前堆栈指针, 寄存器 R15 通常用作链接寄存器存放子程序的返回地址。

ALU 执行标准的 32 位整数操作, 支持快速找 1 算法 (FF1)。

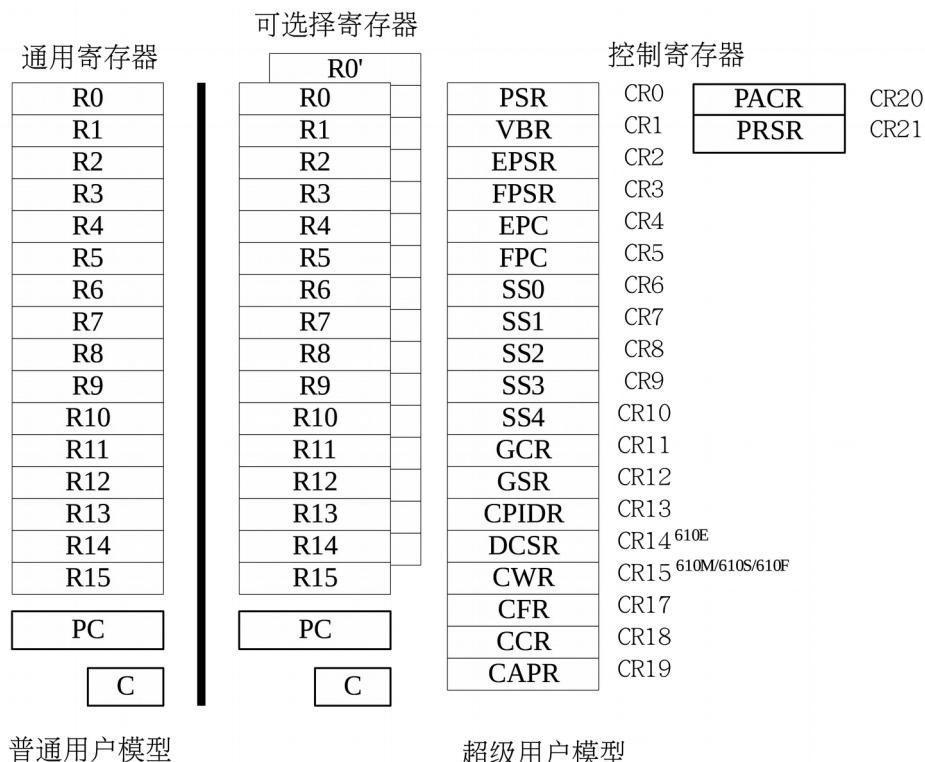
乘法器支持  $16 \times 16$  和  $32 \times 32$  整数乘法, 除法器的设计采用了快速算法, 当除数与被除数大小比较相近时能够大大减少运算所需的时钟数(3~37 个)。

**C-Sky Confidential**

一个条件/进位位（C）用于条件测试和多于 32 位的算术逻辑运算。一般的，C 位只在显式的测试/比较运算中被改变，而不是正常指令运算的副作用。但是在某些特定的把条件设置和运算结合的指令中可能例外。

中断响应快，16 个可选择寄存器用于减少在中断异常处理时花费的时间。支持矢量和自动矢量中断。

### 1.3. 编程模型



图表 简介-2 编程模型

注意：CR14 仅存在于带有 CK610E 特征的 C-SKY CPU 处理器中。

CR15 仅存在于带有协处理器接口的 C-SKY CPU 处理器中（如 CK610M、CK610S、CK610F 等）。

C-SKY CPU 定义了两种编程权限：超级用户模式和普通用户模式。有一些操作在普通用户模式是不允许的。

程序根据其权限来访问寄存器。普通用户程序只允许访问那些指定给普通用户模式的寄存器；工作在超级用户模式下系统软件则可以访问所有的寄存器，使用控制寄存器来进行超级用户操作。普通用户程序因此就可以避免接触那些特权信息，而操作系统通过协调与普通用户程序的行为来为普通用户程序提供管理和服务。

大多数指令在两种模式下都能执行，但是一些对系统产生重大影响的特权指令只能工作在超级用户模式下。比如说，普通用户程序不能执行 stop, doze 或 wait 指令。为防止普通用户程序不受控制地进入超级用户模式，可以改变程序状态寄存器（PSR）S 位的指令也是特权指令。Trap #n 指令为普通用户程序提供了访问操作系统服务程序的可控制接口。在普通用户模式下，访问一些特殊的控制寄存器也是不允许的。

当 PSR 内的 S 位被置位，处理器就在超级用户模式下执行程序。

在普通用户模式下，处理器利用普通用户编程模型。在异常处理时，处理器把模式从普通用户模式切换到超级用户模式。异常处理把当前的 PSR 的值存放在 EPSR 或 FPSR 影子控制寄存器中，然后在 PSR 中设置 S 位，强制处理器进入超级用户模式。在异常处理后，为返回到以前的工作模式，系统函数执行 rte(从异常返回)或 rfi (从快速中断返回)，清空流水线，并从中断发生的地方重新取指执行。

**C-Sky Confidential**

编程模型里描述的寄存器用于存放和控制操作数。所有的寄存器被分成两个访问权限：普通用户模式和超级用户模式。普通用户编程模型包括 16 个 32 位通用寄存器，32 位程序指针（PC）寄存器和条件/进位位（C）。C 位也就是 PSR 的最低位，PSR 中唯一能被在普通用户模式下访问的数据位。超级用户编程模型增加了 16 个 32 位可选择通用寄存器，以及一组控制寄存器和临时寄存器。通常，通用寄存器 R15 用于存放子过程的返回地址，R0 用于当前的堆栈指针。

访问可选择通用寄存器组可以通过设置 PSR 的控制位来实现。对控制寄存器的访问可以用 mfcr 和 mtcr 指令来实现。当 PSR 中的 AF 位置位时，就选中可选择通用寄存器组，处理器从可选择通用寄存器组中取数进行运算；当 AF 位被清零时，就从通用寄存器组中取运算数据。可选择通用寄存器组是用来减少在实时事件处理时在上下文切换上花费的时间。

超级用户编程模型包括含有操作控制和状态信息的 PSR 寄存器，一套用来在异常发生时保存 PSR、PC 的异常影子寄存器和一套用来节省在快速中断中的上下文切换时间的快速中断影子寄存器。

超级用户程序还可以利用 5 个临时寄存器来处理异常事件、一个寄存器保存中断向量表的基址、一个全局状态寄存器和一个全局控制寄存器。

## 1.4. 数据格式

bit31	Big endian				bit0
Byte0	Byte1	Byte2	Byte3		address 0
Byte4	Byte5	Byte6	Byte7		address 4
Byte8	Byte9	ByteA	ByteB		address 8
bit31	Little endian				bit0
Byte3	Byte2	Byte1	Byte0		address 0
Byte7	Byte6	Byte5	Byte4		address 4
ByteB	ByteA	Byte9	Byte8		address 8

图表 简介-3 内存中的数据组织形式

SSSSSSSSSSSSSSSSSSSSSSSSSS	S Byte	单字节有符号
0000000000000000000000000000	Byte	单字节无符号
SSSSSSSSSSSSSSSS	S Halfword	双字节有符号
0000000000000000	Halfword	双字节无符号
Byte0	Byte1	Byte2
		Byte3

图表 简介-4 寄存器中的数据组织结构

C-SKY CPU 支持标准补码的 2 进制整数。每个指令操作数的长度可以明确地编码在程序中（load/store 指令），也可以隐含在指令操作中（index operation, byte extraction）。通常，指令对接收 32 位操作数，产生 32 位结果。

C-SKY CPU 的存储器可以配置成 Big endian 模式或 Little endian 模式。在 Big endian 模式（缺省模式）下，字 0 的最高位字节放在地址 0 上。而在 Little endian 模式下，字 0 的最高位字节放在地址 3 上。在寄存器中，第 31 位是最高位，习惯上，不管是什么 endian 模式，寄存器的 0 字节代表数据最高字节。这个问题只在执行 xtrb[0-3] 时需要考虑。

## 1.5. 操作数寻址能力

C-SKY CPU 通过 load 和 store 指令来访问所有的存储器，在通用寄存器和存储器间交换数据。寄存器 +4 位偏移的寻址模式被 load/store 指令用来寻址字节、半字或字。

C-SKY CPU 实现了一条指令存取多个连续存储单元，stm/lmd 可以在以 R0 所指的存储器地址为基址的连续地址上，存取多到 16 个通用寄存器的值。Stq/ldq 利用寄存器间接寻

**C-Sky Confidential**

址来连续传送 4 个值。

## 1.6. 指令集一览

C-SKY CPU 的指令集是精心设计的，它具有高级语言特性，并为一些频繁执行的指令进行了优化。该指令集包括标准的算术逻辑指令、位操作指令、字节提取指令、数据转移指令、控制流改变指令和条件执行指令，这些条件执行指令有助于减少短跳转的条件转移。

图表 简介-5 将 C-SKY CPU 的指令集按字母顺序排列。第四章将给出更详细的指令说明。

图表 简介-5 C-SKY CPU 的指令集

汇编指令	指令描述
ABS	绝对值
ADDC	进位加
ADDI	立即数加
ADDU	无符号加
AND	按位与
ANDI	立即数按位与
ANDN	按位与非
ASR	算术右移
ASRC	算术右移，更新 C 位
ASRI	立即数算术右移
BCLRI	位清零
BF	条件位为 0 转移
BGENI	立即数位产生
BGENR	寄存器位产生
BKPT	软断点
BMASKI	位屏蔽
BR	无条件跳转
BREV	位取反
BSETI	立即数置位
BSR	子程序位移
BT	条件位为 1 位移
BTSTI	立即数位测试
CLRF	条件位为 0 清零
CLRT	条件位为 1 清零
CMPHS	大于等于比较
CMPLT	小于比较
CMPLTI	立即数小于比较
CMPNE	不相等比较
CMPNEI	立即数不相等比较
CPRC <sup>CP</sup>	读当前协处理器条件位
CPRCR <sup>CP</sup>	读当前协处理器控制寄存器
CPRGR <sup>CP</sup>	读当前协处理器通用寄存器
CPRS <sup>CP</sup>	读当前协处理器状态寄存器
CPWCR <sup>CP</sup>	写数据到当前协处理器控制寄存器
CPWGR <sup>CP</sup>	写数据到当前协处理器的通用寄存器
CPWIR <sup>CP</sup>	写指令到当前协处理器的指令寄存器
CPWSR <sup>CP</sup>	写数据到当前协处理器的状态寄存器
CPSETI <sup>CP</sup>	设置当前运行的协处理器
DEC <sup>F</sup>	条件位为 0 减 1
DEC <sup>G</sup> T	大于减 1
DEC <sup>L</sup> T	小于减 1
DEC <sup>N</sup> E	不等于减 1
DEC <sup>T</sup>	条件位为 1 减 1
DIVS	有符号除法
DIVU	无符号除法
DOZE	休眠
FF1	查找第一个 1
IDLY4	推迟中断响应
INCF	条件位为 0 加 1
INCT	条件位为 1 加 1
IXH	半字索引
IXW	字索引

## C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

JMP JMPI JSR JSRI	跳转 间接跳转 子程序跳转 间接子程序跳转
LD. [BHW] LDM LDQ LRW LSL LSR LSLC LSRC LSLI LSRI	读 RAM 多个读 RAM 4 个读 RAM 间接读 RAM 逻辑左移 逻辑右移 带条件位逻辑左移 带条件位逻辑右移 立即数左移 立即数右移
MAC <sup>620</sup> MFCR MFHI <sup>620</sup> MFLO <sup>620</sup> MFHIS <sup>610E</sup> MFLOS <sup>610E</sup> MOV MOVI MOVF MOVT MTCR MTHI <sup>620</sup> MTLO <sup>620</sup> MULSH MULSHA <sup>610E</sup> MULSHS <sup>610E</sup> MULSWA <sup>610E</sup> MULSWS <sup>610E</sup> MULS <sup>610E</sup> MULSA <sup>610E</sup> MULSS <sup>610E</sup> MULT MULU <sup>610E</sup> MULUA <sup>610E</sup> MULUS <sup>610E</sup> MVC MVCV MVTC <sup>610E</sup> NOT	乘法累加 读控制寄存器 读高位寄存器 读低位寄存器 读高位寄存器并进行饱和运算 读低位寄存器并进行饱和运算 寄存器传送 立即数传送 条件位为 0 传送 条件位为 1 传送 写控制寄存器 写高位寄存器 写低位寄存器 半字有符号乘法 有符号 16 位乘加 有符号 16 位乘减 有符号 16 位和 32 位乘加 有符号 16 位和 32 位乘减 有符号 32 位乘法 有符号 32 位乘加 有符号 32 位乘减 乘法 无符号 32 位乘法 无符号 32 位乘加 无符号 32 位乘减 传送条件位到寄存器 传送条件位的取反到寄存器 传送溢出位到条件位 按位非
OMFLIP <sup>620</sup> OR	基于 omega-flip 网络的总体置换 按位或
ROTLI RSUB RSUBI RTE RFI	立即数循环左移 反向减 反向立即数减 异常返回 快速中断返回
SEXTB SEXTH ST. [BHW] STM STQ STOP SUBC SUBU SUBI SYNC	字节符号位扩展 半字符号位扩展 写 SRAM 多个写 SRAM 4 个写 SRAM 停止模式 带进位减 无符号减 立即数减 同步
TRAP TST TSTNBZ	操作系统陷阱 0 测试 字节 0 测试
VMULSH <sup>610E</sup> VMULSHA <sup>610E</sup>	两路有符号 16 位乘法 两路有符号 16 位乘加

## C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

VMULSHS <sup>610E</sup>	两路有符号 16 位乘减
VMULSW <sup>610E</sup>	两路有符号 16 位和 32 位数乘法
VMULSWA <sup>610E</sup>	两路有符号 16 位和 32 位数乘加
VMULSWS <sup>610E</sup>	两路有符号 16 位和 32 位数乘减
WAIT	等待模式
XOR	按位异或
XSR	扩展右移
XTRB0	提取并扩展字节 0 到 R1
XTRB1	提取并扩展字节 1 到 R1
XTRB2	提取并扩展字节 2 到 R1
XTRB3	提取并扩展字节 3 到 R1
ZEXTB	0 扩展字节
ZEXTH	0 扩展半字

注意：

MAC、MFHI、MFLO、MTHI、MTLO 和 OMFLIP 指令仅存在于 CK620 中。

CPRC、CPRCR、CPRGR、CPRSR、CPWCR、CPWGR、CPWIR、CPWSR 和 CPSETI 指令仅存于带有协处理器接口的处理器中，如 CK610S、CK610M、CK610F 等。

MFHIS、MFLOS、MULSHA、MULSHS、MULSWA、MULSWS、MULS、MULSA、MULSS、MULU、MULUA、MULUS、MVTC、VMULSH、VMULSHA、VMULSHS、VMULSW、VMULSWA 和 VMULSWS 指令仅存在于 CK610E 中。



**C-Sky Confidential**

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

## 2. 命名规则

### 2.1. 符号

本文档用到的标准符号和操作符如下表所示

符号	功能
+	加
-	减
*	乘
/	除
>	大于
<	小于
=	等于
≥	大于或等于
≤	小于或等于
1/4	不等于
.	与
+	或
◆	异或
NOT	取反
:	连接
?	传输
◆	交换
?	误差
0b0011	二进制数
0x0F	十六进制数
610	仅在 CK610 中存在
620	仅在 CK620 中存在
610M	仅在 CK610M 中存在
CP	仅在带有协处理器接口的处理器中存在

### 2.2. 术语

- 逻辑 1 是指对应于布尔逻辑真的电平值。
- 逻辑 0 是指对应于布尔逻辑伪的电平值。
- 置位是指使得某个或某几个位达到逻辑 1 对应的电平值。
- 清除是指使得某个或某几个位达到逻辑 0 对应的电平值。
- 保留位是为功能的扩展而预留的，没有特殊说明时其值为 0。
- 信号是指通过它的状态或状态间的转换来传递信息的电气值。
- 引脚是表示一种外部电气物理连接，同一个引脚可以连接多个信号。
- 使能是指使某个离散信号处在有效状态：
  - 低电平有效信号从高电平切换到低电平；
  - 高电平有效信号从低电平切换到高电平。
- 禁止是指使某个处在使能状态的信号状态改变：
  - 低电平有效信号从低电平切换到高电平；
  - 高电平有效信号从高电平切换到低电平。
- LSB 代表最低有效位,MSB 代表最高有效位。  
存储单元和寄存器当“pad\_biu\_bigend\_b=0”时采用高位收尾模式，其字节次序是高字节在最低位的排列。一个字中的所有位是从最高有效位(第 31 位)开始往下排列。
- 当“pad\_biu\_bigend\_b=1”时，采用低位收尾模式。
- 信号，位域，控制位的表示都使用一种通用的规则。
- 标识符后来跟着表示范围的数字，从高位到低位表示一组信号，比如 addr[4:0]就表示一组地址总线，最高位是 addr[4]，最低位是 addr[0]。

C-Sky Confidential

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).



- 单个的标识符就表示单个信号，例如 pad\_biu\_reset\_b 就表示单独的一个信号。有时候会在标识符后加上数字表示一定的意义，比如 addr15 就表示一组总线中的第 16 位。



**C-Sky Confidential**

The information contained herein is confidential and proprietary and is not to be disclosed outside of Hangzhou C-Sky Microsystems except under a Non-Disclosure Agreement (NDA).

### 3. 寄存器描述

本章主要介绍 C-SKY CPU 通用寄存器和控制寄存器在普通用户模式和超级用户模式下的组织结构。

#### 3.1. 普通用户编程模式

图表 寄存器描述-6 列出了普通用户编程模式下的一些寄存器:

- 16 个 32 位通用寄存器 (R[15:0]) ;
- 32 位程序计数器 (PC) ;
- 条件码／进位标志位 (C bit) 。

名称	功能
R0	堆栈指针
R1	不确定
R2	不确定, 函数调用时第一个参数
R3	不确定, 函数调用时第二个参数
R4	不确定, 函数调用时第三个参数
R5	不确定, 函数调用时第四个参数
R6	不确定, 函数调用时第五个参数
R7	不确定, 函数调用时第六个参数
R8	不确定
R9	不确定
R10	不确定
R11	不确定
R12	不确定
R13	不确定
R14	不确定
R15	链接寄存器

PC	程序计数器
----	-------

C

图表 寄存器描述-6 普通用户编程模式寄存器

##### 3.1.1. 通用寄存器

这些通用寄存器包含了指令操作数和结果以及地址信息。硬软件上惯例就是用这些通用寄存器来做子程序的链接调用，参数传递以及堆栈指针。

##### 3.1.2. 程序计数器

程序计数器包含了当前执行指令的地址。在指令执行和异常处理期间，处理器会根据程序运行的情况自动的调整程序计数器值或放置一新值到程序计数器中。对一些指令来说，程序计数器能被用来作为相对地址计算。此外，程序计数器中的低位一直为零，除非发生不对齐异常。

##### 3.1.3. 条件码／进位标志位

条件码／进位标志位代表了一次操作后的结果。条件码／进位标志位能够作为比较操作指令的结果清楚地被设置，或者作为另一些高精度算术或逻辑指令的结果而不确定地被设置。另外，特殊的指令如 DEC[GT,LT,NE]，以及 XTRB[0-3]等将会影响条件码／进位标志位的值。

### 3.2. 超级用户编程模式

系统编程员用超级用户编程模式来设置系统操作功能，I/O 控制，以及其他受限的操作。

超级用户编程模式由普通用户下的那些寄存器和以下寄存器组成，如图表 寄存器描述-7 所示：

- 16 个 32 位可选择寄存器；
- 处理器状态寄存器(PSR)；
- 向量基址寄存器(VBR)；
- 异常保留程序计数器(EPC)；
- 异常保留处理器状态寄存器(EPSR)；
- 快速中断保留程序计数器(FPC)；
- 快速中断保留处理器状态寄存器(FPSR)；
- 5 个 32 位超级用户寄存器(SS0-SS4)；
- 12 位全控制寄存器(GCR)；
- 12 位全状态寄存器(GSR)；
- 产品序号寄存器(CPIDR)；
- 高速缓存配置寄存器(CCR)；
- 可高缓和访问权限配置寄存器(CAPR)；
- 保护区控制寄存器(PACR)；
- 保护区选择寄存器(PRSR)；

R0'	PSR	CR0	DCSR	CR14 <sup>610E</sup>
R1'	VBR	CR1	CWR	CR15 <sup>CP</sup>
R2'	EPSR	CR2	CFR	CR17
R3'	FPSR	CR3	CCR	CR18
R4'	EPC	CR4	CAPR	CR19
R5'	FPC	CR5	PACR	CR20
R6'	SS0	CR6	PRSR	CR21
R7'	SS1	CR7		
R8'	SS2	CR8		
R9'	SS3	CR9		
R10'	SS4	CR10		
R11'	GCR	CR11		
R12'	GSR	CR12		
R13'	CPI DR	CR13		
R14'				
R15'				

可选择寄存器

控制寄存器

图表 寄存器描述-7 超级用户编程模式附加资源

### 3.2.1. 可选择寄存器

在转向时间关键任务时可选择寄存器可用来减少因当前内容转换和保护现场引起的响应延迟时间。当 PSR (AF) 为 1，则可选择寄存器被选中，所有的指令以前使用通用寄存器现在都将使用可选择寄存器。反之，若 PSR (AF) 为 0，则可选择寄存器不被选中。一些重要的参数和指针值可以保存在这些可选择寄存器中，只要在优先级高的任务执行时可选择寄存器被选中，这些重要的数据就可用。

另外，可选择寄存器中的 R0 作为堆栈指针为任务服务，从而使得独立的堆栈实现起来更有效。

在实际使用中，可选择寄存器也可以在普通用户模式下当 AF 为 1 时被访问。

在异常出现并执行异常服务程序时，异常服务程序矢量入口值的低位将被拷贝到 AF

位中去以用来选择哪一组寄存器。

### 3.2.2. 处理器状态寄存器 (PSR, CR0)

处理器状态寄存器 (PSR) 存储了当前处理器的状态和控制信息，包括 C 位，中断有效位和其他控制位。在超级用户编程模式下，软件可以访问处理器状态寄存器 (PSR)。控制位为处理器指出了以下的状态：跟踪模式 (TM [1:0] )，超级用户模式或者普通用户模式 (S 位)，以及通用寄存器或者可选择寄存器 (AF 位)。它们同样也指出了异常保留寄存器是否可用来保存当前相应的内容，以及中断申请是否有效。

	31	30	28	27	24	22	16
	S	0		CPIID[3:0]		VEC [6: 0]	
Reset	1		0			0	
	15	14	13	12	11	9	8
					10	7	6
		TM[1:0]	TP	TE	0	MM	EE
Reset	0	0	0	0	0	0	0
					0	IC	IE
					0	0	0
					0	FE	0
					0	AF	C
Reset	0	0	0	0	0	0	0

图表 寄存器描述-8 处理器状态寄存器

#### S-超级用户模式设置位:

当 S 为 0 时，处理器工作在普通用户模式；

当 S 为 1 时，处理器工作在超级用户模式；

S 位在被 reset 和进入异常处理时由硬件置 1。

#### CPIID[3:0]当前正在运行的协处理器号:

当硬件上存在多个协处理器单元时，用来表征当前正在运行的协处理器，通过指令 MTCR 可以直接修改该状态段，通过 CPSETI 也可以间接修改该状态端。如果软件配置了硬件上不存在的协处理器号，则该段将会指向软件所配置的协处理器，所有对协处理器的写操作不会发生任何作用，所以对协处理器的读操作，均返回 0。如果 CPIID 被设置成了 0，则所有对协处理器的写操作不会发生任何作用，所以对协处理器的读操作，均返回 0。

#### VEC[6:0]-异常事件向量值:

当异常出现时，这些位被用来计算异常服务程序向量入口地址，且会在被 reset 时清零。

#### TM[1:0]-跟踪模式位:

在指令跟踪模式下，每一条指令执行完后，C-SKY CPU 都将会进入跟踪异常服务程序；在跳转跟踪模式下，当碰到内含有跳转（不管是跳转还是不跳转）的指令执行完，C-SKY CPU 都将会进入跟踪异常服务程序。这些位在被 reset 清零和进入异常服务程序时由硬件清零。图表 寄存器描述-9 示出 TM [1:0] 编码与相对应的工作模式。

值	描述
00	正常执行模式
01	指令跟踪模式
10	未定义
11	跳转跟踪模式

图表 寄存器描述-9 TM[1:0]编码与相对应的工作模式

跟踪模式具体的操作请参考第六章异常处理。

#### TP-待定跟踪异常设置位:

当 C-SKY CPU 工作在跟踪模式（指令跟踪模式或者跳转跟踪模式）下，优先级更高异常和跟踪异常同时发生，C-SKY CPU 会先响应优先级高的异常，从而这跟踪异常将被悬挂而不被处理，但 C-SKY CPU 会将异常保留处理器状态寄存器 (EPSR) 或者快速中断保留处理器状态寄存器 (FPSR) 中的 TP 位设置为 1，以便在优先级高的异常处理完毕时，由 rte 或 rfi 将 EPSR 或者 FPSR 中的 TP 拷贝到 PSR 中，此时 C-SKY CPU 根据 PSR 中的 TP 考虑是否进入跟踪异常服务程序，若此 TP 为 1，则进入跟踪异常服务程序；反之，不进入。PSR 中的 TP 位不能被除了 rte 或 rfi 之外的指令改变，FPSR 和 EPSR 中的 TP 是不能被指令设置的。

**注意：**该位仅仅存在于异常保留处理器状态寄存器 (EPSR) 或者快速中断保留处理器状态寄存器 (FPSR) 中，在处理器状态寄存器 (PSR) 中，该位一直被置 0。

#### **TE-传输控制位:**

该位在芯片的管脚上有对应的管脚信号 biu\_pad\_te\_b，可以通过设置该位来控制传输，它会被 reset 清零，也称为 TC。

#### **MM-不对齐异常掩盖位:**

当 MM 为 0 时，读取或存储的地址不对齐异常将正常发生，不会被掩盖即异常会被响应；

当 MM 为 1 时，读取或存储的地址不对齐异常将会被掩盖，访问内存时读取或存储的地址低位都将会被默认为 0。

该位不能掩盖 jmpi 或 jsri 指令的不对齐异常的发生，且不受其它异常影响，但会被 reset 清零。

#### **EE-异常有效控制位:**

当 EE 为 0 时，异常无效，此时除了普通中断和快速中断之外的任何异常一旦发生，都会被 C-SKY CPU 认为是不可恢复的异常；

当 EE 为 1 时，异常有效，所有的异常都会正常的响应和使用 EPSR 与 EPC。

#### **IC-中断控制位:**

当 IC 为 0 时，中断只能在指令之间被响应；

当 IC 为 1 时，表明中断可在长时间、多周期的指令执行完之前被响应；

会被 reset 清零，不受其它异常影响。

#### **IE-中断有效控制位:**

当 IE 为 0 时，中断无效，即 pad\_biu\_int\_b 不起作用，EPC 和 EPSR 都无效；

当 IE 为 1 时，中断有效，即 pad\_biu\_int\_b 起作用；

该位会被 reset 清零，也在进入异常服务程序时被清零。

#### **FE-快速中断有效控制位:**

当 FE 为 0 时，快速中断无效，即 pad\_biu\_fint\_b 不起作用，FPC 和 FPSR 都无效；

当 FE 为 1（不必考虑 EE 位）时，快速中断有效，即 pad\_biu\_fint\_b 起作用；

该位会被 reset 清零，也在进入快速中断服务程序时被清零，但不受其它异常的影响。

#### **AF-可选择寄存器有效控制位:**

当 AF 为 0 时，通用寄存器被选中，可选择寄存器不被选中；

当 AF 为 1 时，通用寄存器不被选中，可选择寄存器被选中；

当异常发生时，异常入口地址的最低位被拷贝到该位用来选择哪一组寄存器以便在异常服务程序中使用。此位被 reset 清零，但同时它也被 reset 向量的低位所设置。

#### **C-条件码／进位位**

该位用作条件判断位为一些指令服务。它在 reset 和在被拷贝到 EPSR 或 FPSR 之后不确定。

### **3.2.2.1. 更新 PSR**

PSR 可以通过几种不同的方式被更新，对 PSR 中控制位的更改所产生的影响也多种多样。PSR 通常可以通过异常响应，异常处理和执行 psrset, psrclr, rte, rfi, mtcr, cpid 指令被修改，这些修改的实现有四个方面。

#### **● 异常响应和异常处理更新 PSR:**

更新 PSR 是异常响应和异常服务程序入口地址计算中的一部分，它将更新 PSR 中 S, TM, VEC, IE, FE, EE 以及 AF 位。对 S, TM, VEC, IE, FE 以及 EE 位的改动优先于异常服务程序向量入口地址的取址。对 VEC 以及 AF 位的改动优先于异常服务程序中的第一条指令的执行。

#### **● RTE 和 RFI 指令更新 PSR:**

更新 PSR 作为 rte 和 rfi 指令执行的一部分，可能会对 PSR 中的所有位都改动。其中对 S, TM, TP, IE, FE 和 EE 的改动优先于对返回 PC 的取址，对 VEC, MM, IC, AF 和 C 位的改动优先于程序返回后第一条指令的执行。

- MTCR 指令更新 PSR:

若目标寄存器是 CR0 的话，更新 PSR 将会作为 mtcr 指令执行的一部分。这种更新将可能会改变 PSR 中所有位的值，紧接着的指令、异常事件和中断响应将会采用新的 PSR 值。

- PSRCLR、PSRSET 指令更新 PSR:

更新 PSR 作为 psrclr 和 psrset 指令执行的一部分，紧接着的指令、异常事件和中断响应将会采用新的 PSR 值。

### 3.2.3. 向量基址寄存器 (VBR, CR1)

VBR 寄存器用来保存异常向量的基址。该寄存器包含 22 个高位有效位，10 个保留位（其值为 0）。VBR 的复位值为 0X00000000。

31	10	0
Reset	VECTOR BASE	RESERVED
0	0	0

图表 寄存器描述-10 基址向量寄存器

### 3.2.4. 异常保留寄存器 (CR2~CR5)

EPSR, EPC, FPSR 和 FPC 这些寄存器在遇到异常情况时被用来保存当前处理器执行的内容。更详细的信息请参考第六章异常处理。

### 3.2.5. 存储寄存器 SS0~SS4 (CR6~CR10)

C-SKY CPU 包含了 5 个 32 位的超级用户模式下存储寄存器，超级用户可以使用这些寄存器来存储数据，指针，辅助异常处理以及避免受普通用户模式影响。软件确定了这些寄存器的使用的功能和内容。典型的用法是将它们中的一个寄存器用来作为堆栈指针。所有的这些寄存器都可以通过 mfcr 和 mtcrr 来访问并修改其中的内容。

### 3.2.6. 全局控制寄存器 (GCR, CR11)

12 位的全局控制寄存器是用来控制外部设备和事件。它通过芯片口上提供的 12 位平行输出接口实现指定控制。一般来说，可以通过简单设置 GCR 来管理功耗，设备控制，事件安排处理以及其它的基本的功能。至于 GCR 中每一位对应的控制功能，用户可以根据情况自行定义。全控制寄存器是可读可写的。

### 3.2.7. 全局状态寄存器 (GSR, CR12)

12 位的全局状态寄存器是用来标记外围设备和事件的。它通过芯片口上提供的 12 位平行输入接口将外部状态送入到 C-SKY CPU 内部，从而实现监测。一般来说，可以通过查看 GSR 来检测外围设备状态和事件。全状态寄存器是只读的。

### 3.2.8. 产品序号寄存器 (CPUIDRR, CR13)

该寄存器用于存放杭州中天微系统有限公司产品的内部编号。产品序号寄存器是只读的，其复位值由产品本身决定。

31	28	27	24	23	20	19	16	15	12	11	8	7	4	3	0
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

图表 寄存器描述-11 产品序号寄存器

31	28	27	24	23	20	19	16	15	12	11	10	9	8	7	4	3	0
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

**Family-产品系列：**

当 Family 为 0100 时, CK6XX 系列产品。  
其余保留。

**Model-产品模型:**

当 Model 为 0000 时, CK610;  
当 Model 为 0001 时, CK620;  
当 Model 为 0010 时, CK660;  
当 Model 为 0011 时, CK610e;  
其余保留。

**IC-指令高缓大小:**

当 IC 为 0000 时, 无指令 Cache;  
当 IC 为 0001 时, 2K 指令 Cache;  
当 IC 为 0010 时, 4K 指令 Cache;  
当 IC 为 0011 时, 8K 指令 Cache。  
其余保留。

**DC-数据高缓大小:**

当 DC 为 0000 时, 无数据 Cache;  
当 DC 为 0001 时, 2K 数据 Cache;  
当 DC 为 0010 时, 4K 数据 Cache;  
当 DC 为 0011 时, 8K 数据 Cache。  
其余保留。

**Boundary-制造商:**

当 Boundary 为 0000 时, TSMC;  
当 Boundary 为 0001 时, SMIC;  
当 Boundary 为 0010 时, Hejian;  
当 Boundary 为 0011 时, HHNEC。  
其余保留。

**ISPM-硬件支持 ISPM:**

当 ISPM 为 1 时, 硬件上支持 ISPM;  
当 ISPM 为 0 时, 硬件上不支持 ISPM。

**DSPM-硬件支持 DSPM:**

当 DSPM 为 1 时, 硬件上支持 DSPM;  
当 DSPM 为 0 时, 硬件上不支持 DSPM。

**Process-制造工艺:**

当 Process 为 00 时, 0.18um;  
当 Process 为 01 时, 0.13um。  
其余保留。

### 3.2.9. DSP 控制状态寄存器<sup>610E</sup> (DCSR, CR14)

CK610E 提供一个额外的控制状态寄存器来控制 CK610E 所支持的 DSP 指令的执行, 同时用来表征这些 DSP 指令执行结果的是否出现溢出。

31	FM	0	V	0
----	----	---	---	---

图表 寄存器描述-12 DSP 控制状态寄存器

**FM-小数运算模式:**

当 FM 位被设置成 1 时, 所有的乘法运算被认为是小数运算。

**V-溢出状态位:**

用来表征最近的一次乘加或者乘减操作是否溢出。这一位为只读位。通过使用 MVTC 指

令可以将这一比特位拷贝到 PSR 的状态位 (C 位)。

### 3.2.10. 协处理器窗口寄存器<sup>CP</sup> (CPWR, CR15)

用户通过读写协处理器窗口寄存器来控制当前协处理器的执行状态。每一个协处理器都有自己的协处理器窗口寄存器。用户通过 CPSETI 指令或者 MFCR 指令设置完成当前所需要配置的协处理器后，通过配置此窗口寄存器来控制当前协处理器的运行状态。

31	30	29	28	27	24	23	3	2	1	0
IE	0	IC	EC		EP		0	RP	WP	EN

图表 寄存器描述-13 协处理器窗口寄存器

#### IE-中断使能位:

当 IE 位为 0 时，当前协处理器所产生中断的信号将会被屏蔽；

当 IE 位为 1 时，当前协处理器所发生中断的信号将会被使能。

#### IC-清除中断位:

当 IC 位为 1 时，表示清除当前协处理器所发生的中断。

#### EC-清除异常位:

当 EC 位为 1 时，表示清除当前协处理器所发生的异常。

#### EP-设置中断优先级:

通过设置 EP 来设置当前协处理器的中断优先级，设置的值从 1 (最高优先级) 到 15 (最低优先级)。

#### RP-CPREAD 指令精确异常模式位:

通过设置该比特位来控制处理器在执行 CPREAD 指令时，是否处于精确异常模式。

1 表示处于精确异常模式。

#### WP-CPWRITE 指令精确异常模式位:

通过设置该比特位来控制处理器在执行 CPWRITE 指令时，是否处于精确异常模式。

1 表示处于精确异常模式。

#### EN-当前协处理器使能位:

当此比特位为 1 时，表示使能当前的协处理器

### 3.2.11. 高速缓存功能设置寄存器 (CFR, CR17)

高速缓存功能设置寄存器用来控制高速缓存，让相应高速缓存内的数据全部无效。

31	30	8	7	6	5	4	3 2	1	0
Reset	LICF	0	ITS	OMS	CLR	INV	0	CACHE_SEL	0

图表 寄存器描述-14 高速缓存功能设置寄存器

#### LICF-Cache Line INV/CLR 失败状态位:

当一个使用虚拟地址作为索引的 Cache line INV/CLR 操作，在虚拟地址转换的过程中遇到 TLB MISS/TLB FATAL/TLB READ INV/ACCESS ERR 等异常时，此位将被置起。

#### ITS-Cache INV/CLR 模式位:

当 ITS 为 0 时，CIR 将被用于虚拟地址索引。

当 ITS 为 1 时，CIR 将被用于 WAY/SET 索引。

#### OMS-Cache INV/CLR 模式位:

当 OMS 为 0 时，cache 的 CLR 和 INV 操作将对 cache 的所有 line 起作用。

当 OMS 为 1 时，cache 的 CLR 和 INV 操作将对 cache 的一个 line 起作用。并且使用 CIR 作为 cache INV/CLR 操作的索引。

#### CLR-脏表项清除设置位:

当 CLR 为 1 时，高速缓存内的被标记为脏的表项中的数据将被写出到片外内存中。

#### **INV-无效设置位:**

当 INV 为 1 时，高速缓存内的数据将失效。

#### **CACHE\_SEL-高速缓存选择位:**

当 SEL 为 01 时，选中指令高速缓存；

当 SEL 为 10 时，选中数据高速缓存；

当 SEL 为 11 时，选中数据和指令高速缓存。

### **3.2.12. 高速缓存配置寄存器 (CCR, CR18)**

高速缓存配置寄存器用来配置内存，高速缓存有效／无效，内存保护区，Endian 模式，以及系统和处理器的时钟比。

Reset	0	SCK	BE	Z	RS	WB	DE	IE	MP
	0	-	-	0	0	0	0	0	0

图表 寄存器描述-15 高速缓存配置寄存器

#### **SCK-系统和处理器的时钟比:**

该位用来表示系统和处理器的时钟比，其计算公式为：时钟比 = SCK + 1，CPU 上有对应引脚引出。SCK 在 reset 时被配置且不能在之后改变。

#### **BE-Endian 模式:**

当 BE 为 0 时，Little endian；

当 BE 为 1 时，Big endian；

BE 在 reset 时被配置且不能在之后改变，CPU 上有对应引脚引出。

#### **Z-允许预测跳转设置位:**

当 Z 为 0 时，预测跳转关闭；

当 Z 为 1 时，预测跳转开启。

#### **RS-地址返回栈设置位:**

当 RS 为 0 时，返回栈关闭；

当 RS 为 1 时，返回栈开启。

#### **WB-高速缓存写回设置位:**

当 WB 为 0 时，数据高速缓存为直写模式；

当 WB 为 1 时，数据高速缓存为写回模式。

#### **DE-数据高速缓存设置位:**

当 DE 为 0 时，数据高速缓存关闭；

当 DE 为 1 时，数据高速缓存开启。

#### **IE-指令高速缓存设置位:**

当 IE 为 0 时，指令高速缓存关闭；

当 IE 为 1 时，指令高速缓存开启。

#### **MP-内存保护设置位:**

在没有 MMU 功能单元的处理器如 CK610 中，MP 用来设置 MGU 是否有效，如下表：

MP	功能
00	MGU 无效
01	MGU 有效
10	MGU 无效
11	MGU 有效

图表 寄存器描述-16 CK610, CK620 内存保护设置

#### **MP-内存保护设置位:**

在含有 MMU 功能单元的处理器如 CK610M 中，MP 用来设置 MMU 是否有效，如下表：

MP	功能
00	MMU 无效
01	MMU 有效

10	MMU 无效
11	MMU 有效

图表 寄存器描述-17 CK610M 内存保护设置

### 3.2.13. 可高缓和访问权限配置寄存器 (CAPR, CR19)

CAPR 的各位如下图所示:

31	16	15	14	13	12	11	10	9	8	7	4	3	2	1	0
Reset	0	AP3		AP2		AP1		AP0	0	0	0	C3	C2	C1	CO
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

图表 寄存器描述-18 可高缓和访问权限配置寄存器

#### C0~C3-可高缓属性设置位:

当 C 为 0 时, 该区不可以进行高速缓存操作;

当 C 为 1 时, 该区可以进行高速缓存操作。

#### AP0~AP3-访问权限设置位:

AP	超级用户权限	普通用户权限
00	不可访问	不可访问
01	读写	不可访问
10	读写	只读
11	读写	读写

图表 寄存器描述-19 访问权限设置

### 3.2.14. 保护区控制寄存器 (PACR, CR20)

PACR 的各位如下图所示:

31	12	11	6	5	1	0
Reset	Base Address		0	Size	E	
	-		0	-	0	

图表 寄存器描述-20 保护区控制寄存器

#### Base Address-保护区地址的高位:

该寄存器指出了保护区地址的高位, 但写入的基址必须与设置的页面大小对齐, 例如设置页面大小为 8M, CR20[22:12]必须为 0, 各页面的具体要求见下表: 图表 寄存器描述-21 保护区大小配置和其对基址要求。

#### Size-保护区大小:

保护区大小从 4KB 到 4GB, 它可以通过公式: 保护区大小 =  $2^{(Size+1)}$  计算得到。因此 Size 便从 01011 到 11111, 其它一些值都会造成不可预测的结果。

Size	保护区大小	Base Address Requirement
00000—01010	保留	—
01011	4KB	没有要求
01100	8KB	CR20.bit[12]=0
01101	16KB	CR20.bit[13:12]=0
01110	32KB	CR20.bit[14:12]=0
01111	64KB	CR20.bit[15:12]=0
10000	128KB	CR20.bit[16:12]=0
10001	256KB	CR20.bit[17:12]=0
10010	512KB	CR20.bit[18:12]=0
10011	1MB	CR20.bit[19:12]=0
10100	2MB	CR20.bit[20:12]=0
10101	4MB	CR20.bit[21:12]=0
10110	8MB	CR20.bit[22:12]=0
10111	16MB	CR20.bit[23:12]=0
11000	32MB	CR20.bit[24:12]=0
11001	64MB	CR20.bit[25:12]=0
11010	128MB	CR20.bit[26:12]=0
11011	256MB	CR20.bit[27:12]=0
11100	512MB	CR20.bit[28:12]=0
11101	1GB	CR20.bit[29:12]=0

11110	2GB	CR20.bit[30:12] =0
11111	4GB	CR20.bit[31:12] =0

图表 寄存器描述-21 保护区大小配置和其对基址要求

#### E-保护区有效设置:

- 当 E 为 0 时，保护区无效；
- 当 E 为 1 时，保护区有效。

### 3.2.15. 保护区选择寄存器 (PRSR, CR21)

PRSR 用来选择当前操作的保护区，其各位如下图所示：

31	0	2	1	0
Reset	0	-	RID	-

图表 寄存器描述-22 保护区选择寄存器

#### RID-保护区索引值:

RID 可以是 00, 01, 10 或者 11，分别选择第一，二，三或者四保护区。

### 3.2.16. 高速缓存索引寄存器 (CIR, CR22)

当 CFR 的 ITS 被配置为 0 时，CIR 的各个位如下

31	0	0
Reset	虚拟地址	0

当 CFR 的 ITS 被配置为 1 时，CIR 的各个位如下

31	30	15	14	4	3	0
Reset	way	-	0	set	-	0

Set: 指示操作的块位于高速缓存特定路的第几行。

Way: 指示操作的块位于高速缓存第几路。

### 3.2.17. CPU HINT 寄存器 (CHR, CR<30,0>)

31	4	4	3	2	1	0
Reset	0	LRU	MB	PLD	0	0

图表 寄存器描述-23 CPU HINT 寄存器

#### PLD-读预取加速有效设置:

该位有效时，CPU 在检测到连续的 cache line 的缺失时，会自动装载接下来的 cache line；此功能可用于加速内存拷贝的性能。

#### MB-写合并有效设置:

该位有效时，CPU 在检测到对于顺序 32 个字范围内的写操作时，会自动将这些写操作合并成一次总线的突发传输，以提升总线的吞吐量；注意此功能的开启需要系统的 slave 支持 strb 传输。

#### LRU-MMU 采用 LRU 替换策略:

该位有效时，MMU 的第一级 TLB 将采用 LRU（最近最少使用）替换策略，此功能可以提高 TLB 的性能。

### 3.2.18. MGU 使用操作

#### 3.2.18.1. MGU 有效控制

CR20 中的第 0 位是 MGU 有效控制位。在 MGU 有效之前，至少有一个区被指定以及它相应的 C 和 AP 也必须被设置。此外，这个让 MGU 有效的指令必须在指令地址访问有效的

范围之内，即此指令所在的区域不可以在 MGU 中设置为拒绝访问。若不这么做，将会导致不可预测的结果，可能出现悬挂状态，这样的话只能通过系统的 reset 来恢复了。当 MGU 无效的时候，所有内存的访问都认为是不可高缓的，也不会出现中途失败（abort 信号有效）。

### 3.2.18.2. 内存访问处理

在内存访问信号产生之后，MGU 会检查当前访问的地址是否在这四个保护区内：  
如果访问的地址不在四个区中的任何一个，此内存访问会中途停止；  
如果访问的地址在四个区中的一个或多个内，此访问被已使能的最高索引区（3 为最高，0 为最低）所控制。

### 3.2.18.3. 内存访问起始地址设置

CR20 中定义了四个保护区的起始地址和大小。保护区大小必须是 2 的幂，能从 4KB 到 4GB。起始地址必须与区大小对齐，比如：8KB 大小的保护区，起始地址可以是 32'h12346000，但是 16KB 大小的保护区，起始地址就不可以为这个值，可以是 32'h12344000。

## 3.2.19. MMU 使用操作

C-SKY CPU 的虚拟内存地址空间可运行在两个权限级别上：普通用户模式和超级用户模式，如下图左所示。普通用户只能访问 0x00000000~0x7FFFFFFF 表示的 2GB 空间，在访问 0x80000000~0xFFFFFFFF 表示的 2GB 空间时，出现访问错误。

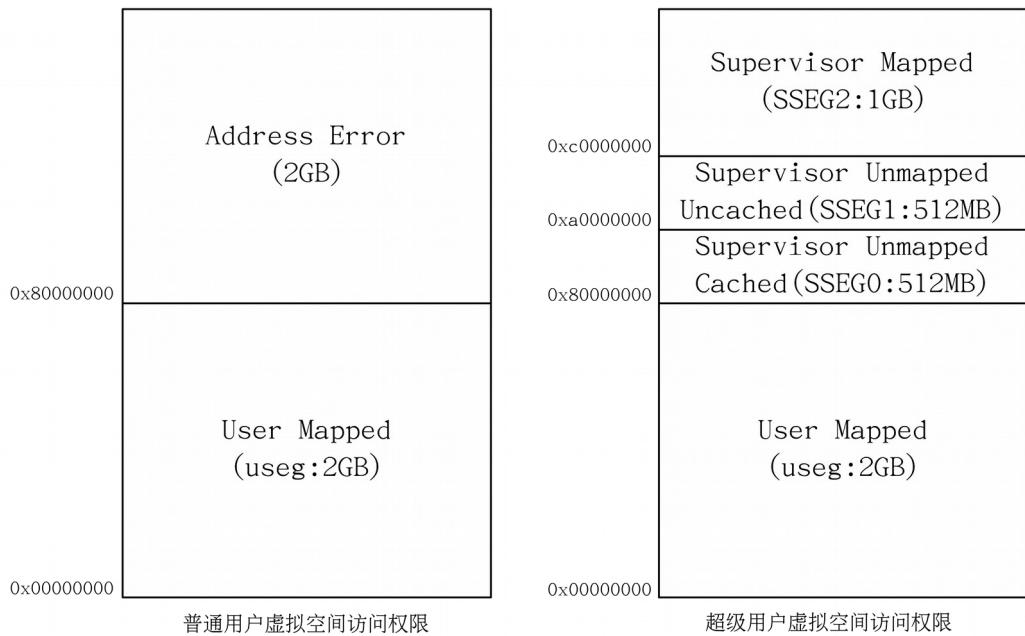
从虚拟地址映射的角度，4GB 空间被划分为 4 个区：

**USEG** (0x00000000~0x7FFFFFFF)：2GB 用户模式下的虚拟地址访问空间。用户在 MMU 被初始化之前不能使用这个区域。

**SSEG0** (0x80000000~0x9FFFFFFF)：512Mbyte 内核模式下的不可映射、可 Cache 空间。这个空间不可映射指在默认情况下，落入这个区的虚拟地址会被强制减掉 0x80000000 后作为物理地址直接返回（即物理地址是通过虚拟地址减去 0x80000000 获得）；用户也可以通过配置 MSA0 来设置映射的物理地址，但是必须是 512MB 页面对齐。

**SSEG1** (0xA0000000~0xBFFFFFFF)：512Mbyte 内核模式下的不可映射、不可 Cache 空间。这个空间的不可映射指在默认情况下，落入该区域的虚拟地址会被强制减掉 0xA0000000 后作为物理地址直接返回（即物理地址是通过虚拟地址减去 0xA0000000 获得）。由于它的不可 Cache 属性，该区通常作为外设的映射空间；用户也可以通过配置 MSA1 来设置映射的物理地址，但是必须是 512MB 页面对齐。

**SSEG2** (0xC0000000~0xFFFFFFFF)：1GB 内核模式下的可映射、可 Cache 空间。这个空间允许通过 MMU 的 TLB 进行物理地址的灵活映射。



CK610M 通过一个协处理器（CP15）来实现内存管理单元（MMU）的功能。系统程序员在超级用户编程模式下来通过设置与 MMU 相关的状态寄存器，来进行 MMU 的管理和操作。这些寄存器包括：

- MMU 索引寄存器<sup>610M</sup> (MIR);
- MMU EntryLo0 寄存器<sup>610M</sup> (MEL0);
- MMU EntryLo1 寄存器<sup>610M</sup> (MEL1);
- MMU EntryHi/Bad VPN 寄存器<sup>610M</sup> (MEH);
- MMU 上下文寄存器<sup>610M</sup> (MCR);
- MMU 页掩码寄存器<sup>610M</sup> (MPR);
- MMU 控制指令寄存器<sup>610M</sup> (MCIR);
- MMU PGD 基址及硬件回填使能控制寄存器 (MPGD)；
- MMU SSEG0 区域配置寄存器 (MSA0)；
- MMU SSEG1 区域配置寄存器 (MSA1)。

### 3.2.20. MMU 索引寄存器<sup>610M</sup> (MIR, CP15\_CR0)

MIR 寄存器用于指示读写操作的 TLB 表项，或存放 TLBP 指令匹配的结果，其各位如下图所示：

	31	30	29	10	9	0
Reset	P	TF	Reserved		Index	
	0	0	0		0	

图表 寄存器描述-24 MMU 索引寄存器

#### P - 匹配失败标志：

当执行 TLBP 指令时，硬件写该位指示 jTLB 是否匹配成功。

当 P 为 0 时，jTLB 匹配成功；

当 P 为 1 时，jTLB 匹配失败。

#### TF (tfatal) TLB 不可恢复异常标志：

该位用于表征是否出现 TLB 不可恢复异常；当 TF 为 1 时，表示出现 TLB 不可恢复异常。

#### Index 索引：

软件写该域以提供 TLB 表项的索引，参考 TLBR 和 TLBWI 用法说明。

当执行 TLBP 指令时，硬件用匹配的 jTLB 表项的索引写该域。如果 TLBP 指令匹配失败，该域的内容不可预知。

新版 CK610M，jTLB 表可配置为 64 或 128 个表项，vTLB 固定为 4 个表项，jTLB 表项的索引号为 0 至 63/127，vTLB 表项的索引号固定为 512、513、514、515。

### 3.2.21. MMU 随机寄存器<sup>610M</sup>(MRR, CP15\_CR1)

C-SKY CPU CK610 系列处理器不再实现此寄存器。

### 3.2.22. MMU EntryLo0 和 EntryLo1 寄存器<sup>610M</sup>(MEL0 & MEL1, CP15\_CR2 & CP15\_CR3)

MEL 寄存器保存 jTLB 访问的物理地址高位和页面属性信息，其各位如下图所示：

31	26	25	6	5	3	2	1	0
Reserved		PFN	C	D	V	G		
Reset	-	-	-	-	-	-	-	-

图表 寄存器描述-25 MMU EntryLo0 和 EntryLo1 寄存器

#### PFN-页帧号：

该域对应于物理地址的 31 位到 12 位。

#### C-可高缓位：

指示当前页是否可高缓：

当 C 为 010 时，当前页不可高缓；

当 C 为 011 时，当前页可高缓；

其他：保留。使用保留值，处理器的行为不可预知。

#### D-Dirty 位：

指示当前页是否可写。

当 D 为 1 时，当前页可写；

当 D 为 0 时，当前页不可写，写该页将产生 jTLB 修改异常。

#### V-有效位：

指示 jTLB 表项虚拟页映射是否有效。

当 V 为 1 时，该页有效，可访问；

当 V 为 0 时，该页无效，不可访问，访问该页将产生 jTLB 无效异常。

#### G-Global 位：

jTLB 写时，MEL0 和 MEL1 寄存器的 G 位的逻辑与作为 jTLB 表项的 G 位。

当 jTLB 表项 G 位为 1 时，在 jTLB 匹配过程中不比较 ASID。

### 3.2.23. MMU EntryHi/Bad VPN 寄存器<sup>610M</sup>(MEH, CP15\_CR4)

MEH 寄存器有两个功能：包含 jTLB 访问的虚拟地址信息和 jTLB 异常时当前 VPN 的值。ASID 表示当前页面对应的进程号。其各位如下图所示：

31	12	11	8	7	0
VPN		Reserved		ASID	
Reset	-	-	-	-	-

图表 寄存器描述-26 MMU EntryHi/Bad VPN 寄存器

#### VPN-虚拟页序号：

该域在 jTLB 读和 TLB 异常时由硬件写，在软件写 jTLB 表项之前由软件预先写入。

注意：在 jTLB 访问时，该域的最低位无意义。在 TLB 异常时，该域的最低位有意义。软件将通过该位分辨哪个页面出现异常。例如，当产生 jTLB 修改异常时，软件将通过该位找到对应的发生修改异常的页面，然后设置相应的 Dirty 位。

### ASID-地址空间标识:

该域通常用来保存操作系统所看到的当前地址空间的标识，异常不会改变其值，因此在jTLB 异常发生后，它依然保存着当前运行进程的正确地址空间标识。

绝大部分软件系统会把当前进程的地址空间标识写入该域。在使用 TLBR 指令读取jTLB 表项时必须要小心，该操作会覆盖整个 MEH 寄存器，因此在执行该操作之后必须恢复当前进程地址空间的标识。

在软件读写 jTLB 表项时，该域用于指示当前表项所属的进程号；在 MMU 进行地址映射时，该域用于指示当前进程号。

该域在软件读 jTLB 表项时由硬件设置；在软件写 jTLB 表项时由软件设置。

### 3.2.24. MMU 上下文寄存器<sup>610M</sup>(MCR, CP15\_CR5)

MCR 寄存器包含PTE(page table entry)中一个表项的指针(虚拟地址)。PTE是操作系统存储虚拟地址到物理地址映射关系的数据结构。在TLB失配异常发生后，操作系统从PTE装载传输失配的jTLB。其各位如下图所示：

31	22 21	3 2	0
	PTEBase	BADVPN2	Reserved
Reset	-	-	-

图表 寄存器描述-27 MMU 上下文寄存器

#### PTEBase-页表表项基址:

该域被操作系统使用，当发生TLB失配异常时，操作系统可以用该域和BADVPN2域的内容迅速找到失配的PTE表项。

#### BADVPN2-失配虚拟页面号:

该域在TLB失配异常时由硬件写入，包含产生异常的虚拟地址的31位到13位。

### 3.2.25. MMU 页掩码寄存器<sup>610M</sup> (MPR, CP15\_CR6)

MPR 寄存器用于配置页面的大小，其各位如下图所示：

31	25 24	13 12	0
	Reserved	Page Mask	Reserved
Reset	-	0	-

图表 寄存器描述-28 MMU 页掩码寄存器

#### Mask-页掩码:

该域的位为 1 表示虚拟地址的对应位不用于 jTLB 匹配。

页掩码的编码如表 3 所示：

页掩码	页大小	奇偶块选择位
0000_0000_0000	4KB	VAddr[12]
0000_0000_0011	16KB	VAddr[14]
0000_0000_1111	64KB	VAddr[16]
0000_0011_1111	256KB	VAddr[18]
0000_1111_1111	1MB	VAddr[20]
0011_1111_1111	4MB	VAddr[22]
1111_1111_1111	16MB	VAddr[24]

图表 寄存器描述-29 页掩码编码

如果软件装载到 MPR 寄存器的值与上面列出的值都不一样，处理器的操作将不可预知。

### 3.2.26. MMU 圈连寄存器<sup>610M</sup>(MWR, CP15\_CR7)

C-SKY CPU CK610 系列处理器不再实现此寄存器。

### 3.2.27. MMU 控制指令寄存器<sup>610M</sup>(MCIR, CP15\_CR8)

MCIR 寄存器实现支持操作 MMU 的命令，包括 jTLB 查找、jTLB 读、jTLB 写索引、jTLB 写随机、jTLB 无效。这些操作可以通过 MTCR 指令实现。其各位如下图所示：

	31	30	29	28	27	26	25	24	8	7	0
Reset	TLBP	TLBR	TLBWI	TLBWR	TLBINV	TLB INV_ALL	TLB INV_IDX	Reserved		ASID	
0	0	0	0	0	0	0	0	0		0	

图表 寄存器描述-30 MMU 控制指令寄存器

#### TLBP-jTLB 查找：

该控制指令用于在jTLB中查找虚拟页号和ASID跟MEH寄存器中内容相匹配的表项，并把相应表项的索引值存入MIR寄存器。如果匹配失败，MIR寄存器的P位被置1。

#### TLBR-jTLB 读：

该控制指令用于从jTLB读一个表项。由MIR指向的jTLB表项的内容被装载到MEH、MEL0、MEL1和MPR寄存器。如果MIR寄存器的值大于或等于jTLB表项的数量，处理器的操作将不可预知。

#### TLBWI-jTLB 写索引：

该控制指令用于写MIR寄存器指向的jTLB表项，即将MEH、MEL0、MEL1和MPR寄存器的内容写入到由MIR寄存器指向的jTLB表项。如果MIR寄存器的值大于或等于jTLB表项的数量，处理器的操作将不可预知。

#### TLBWR-jTLB 写随机：

该控制指令用于写MRR寄存器指向的jTLB表项，即将MEH、MEL0、MEL1和MPR寄存器的内容写入到由MRR寄存器指向的jTLB表项。

#### TLBINV-jTLB 无效：

该控制指令用于删除与ASID域值相同的所有jTLB表项。

#### TLBINV\_ALL-TLB 无效全部表项：

该控制指令用于删除与jTLB和vTLB中的所有TLB表项。

#### TLBINV\_IDX-TLB 无效制定表项：

该控制指令用于删除与MIR寄存器指定的index号所对应的TLB表项。

#### ASID-再循环 ASID：

该域仅仅对jTLB无效指令有意义。如果存在多于256个进程，可能需要再循环ASID。

### 3.2.28. MMU PGD 基址寄存器(MPGD, CP15\_CR29)

MPGD寄存器包含PGD\_BASE\_address（物理地址）和硬件回填使能控制位。在TLB缺失且硬件回填使能时，不产生TLB失配异常，操作系统从内存中读取回填失配的TLB。其各位如下图所示：

	31	12	11	1	0
Reset	PBA	Reserved	Reserved	HRE	
0	0	0	0	0	

图表 寄存器描述-31 MMU PGD 基址寄存器

#### PBA-PGD 表项基地址：

此为PGD表的基地址，为物理地址。

#### HRE-硬件回填使能位：

此为硬件回填使能控制位，当有效时TLB发生了失配，MMU单元会通过BIU单元到内存中读取相关表项。

### 3.2.29. MMU SSEG0 配置寄存器(MSA0, CP15\_CR30)

MSA0寄存器配置SSEG0区映射的物理地址和该区属性的信息，其各位如下图所示：

31	29	28	4	3	2	1	0
BA	Reserved			C	D	V	0

Reset	0	0	0	0	0	0	0
图表 寄存器描述-32 MMU EntryLo0 和 EntryLo1 寄存器							

#### BA-SSEG0 映射的物理地址:

该域对应于物理地址的 31 位到 29 位。

#### C-可高缓位:

指示 SSEG0 区是否可高缓:

当 C 为 0 时, 当前页不可高缓;

当 C 为 1 时, 当前页可高缓;

#### D--脏 (Dirty) 位:

指示 SSEG0 区是否可写。

当 D 为 1 时, 当前页可写;

当 D 为 0 时, 当前页不可写, 写该页将产生 TLB 修改异常。

#### V-有效位:

指示 SSEG0 区映射是否有效。

当 V 为 1 时, 该区有效, 可访问;

当 V 为 0 时, 该区无效, 不可访问, 访问该页将产生 TLB 无效异常。

### 3.2.30. MMU SSEG1 配置寄存器(MSA1, CP15\_CR31)

MSA0 寄存器配置 SSEG1 区映射的物理地址和该区属性的信息, 其各位如下图所示:

	31	29	28	4	3	2	1	0
Reset	BA		Reserved		C	D	V	0
	0		0		0	0	0	0

图表 寄存器描述-33 MMU EntryLo0 和 EntryLo1 寄存器

#### BA-SSEG1 映射的物理地址:

该域对应于物理地址的 31 位到 29 位。

#### C-可高缓位:

指示 SSEG1 区是否可高缓:

当 C 为 0 时, 当前页不可高缓;

当 C 为 1 时, 当前页可高缓;

#### D-脏 (Dirty) 位:

指示 SSEG1 区是否可写。

当 D 为 1 时, 当前页可写;

当 D 为 0 时, 当前页不可写, 写该页将产生 TLB 修改异常。

#### V-有效位:

指示 SSEG1 区虚拟页映射是否有效。

当 V 为 1 时, 该页有效, 可访问;

当 V 为 0 时, 该页无效, 不可访问, 访问该页将产生 TLB 无效异常。

### 3.2.31. jTLB 表项结构 610M

jTLB 包含两部分: 比较部分和物理转换部分。比较部分包括表项的虚拟页序号 (因为每个表项都映射到两个物理页, 所以事实上是虚拟页序号 /2, VPN2), 压缩的页掩码, ASID 和 G 位。物理转换部分包括一对表项, 每个表项都包含物理页帧序号(PFN), 有效(V)位, dirty (D)位和可高缓域(C)。每个 jTLB 表项对应两个物理页面并包含一个表项有效位。jTLB 表项的位定义如下图所示:

83	65	64	59	58	51	50	49	30	29	25	24	5	4	0
VPN2	Compressed		ASID		G		PFN1		Flags1		PFN0		Flags0	

图表 寄存器描述-34 jTLB 表项

压缩的页掩码编码如下表所示:

压缩的页掩码	页掩码	页大小	奇偶块选择位
00_00_00	0000_0000_0000	4KB	VAddr[12]
00_00_01	0000_0000_0011	16KB	VAddr[14]
00_00_11	0000_0000_1111	64KB	VAddr[16]
00_01_11	0000_0011_1111	256KB	VAddr[18]
00_11_11	0000_1111_1111	1MB	VAddr[20]
01_11_11	0011_1111_1111	4MB	VAddr[22]
11_11_11	1111_1111_1111	16MB	VAddr[24]

图表 寄存器描述-35 压缩的页掩码编码

### 3.2.32. 高速暂存器 (SPM)

C-SKY CPU CK610 系列处理器采用指令高速缓存 (cache) 和数据高速缓存分离的内存结构，极大地提升了处理器的性能。在一些应用领域，存储在高速缓存中的某些特殊的指令或数据，会被软件频繁的访问。由于高速缓存对软件是透明的，这些特殊的指令或数据可能会被动态的替换出高速缓存。如果一种片上缓存器，可以常驻这些特殊的指令或数据，那么对于提高软件的执行效率会非常有意义。

高速暂存器 (SPM) 就是这样一种片上缓存器。它有如下一些特性：

- 高速读写指令/数据，只有一个时钟的延迟；
- 高速缓存的兼容性，通过配置成 cache 模式，采用与高速缓存相同的替换策略；
- 通过片上 DMA，可以实现与片外数据的交互；
- 高速缓存模式 (cache 模式) 和片上内存模式 (local memory 模式) 动态可配；
- 为节省总线带宽，数据高速暂存器 (Data SPM) 在内存模式不再支持写直方式；

CK610S 通过一个协处理器 (CP15) 来实现高速暂存器 (SPM) 的功能。系统程序员在超级用户编程模式下来通过设置/读取与 SPM 相关的控制/状态寄存器，来进行 MMU 的管理和操作。这些寄存器包括：

- 指令 SPM 寄存器<sup>610S</sup>；
- 数据 SPM 寄存器<sup>610S</sup>；
- DMA 使能寄存器<sup>610S</sup>；
- DMA 控制寄存器<sup>610S</sup>；
- DMA 状态寄存器<sup>610S</sup>；
- DMA 外部起始地址寄存器<sup>610S</sup>；
- DMA 内部起始地址寄存器<sup>610S</sup>；
- DMA 传输大小寄存器<sup>610S</sup>。

### 3.2.33. 指令 SPM 寄存器(CP15\_CR9)

此寄存器用以描述指令 SPM 的物理地址基址 (base address)。由于 CK610S 现在只支持一块指令 SPM，所以此寄存器只支持一块内存空间的映射。

31	12	11	8	7	4	3	2	1	0
Base Address		Reserved		Size		Reserved	MS	EN	

图表 寄存器描述-36 指令 SPM 寄存器

复位值：0x0000\_00X0

X 表示 SPM 的大小

[31:12]	Base Address	读写	描述指令 SPM 的基址 (base address)。此地址为物理地址。用户在配置基址时，需要按照 SPM 块的起始地址对齐此基址。例如：配置 8KB 大小的 SPM，基址必须设置成 20'b?????????????????????0。 复位值：0
[11:8]	Reserved	只读	全零。 表示 SPM 大小： 4'b0000 0K 4'b0010 4K (最小) 4'b0011 8K 4'b0100 16K 4'b0101 32K 4'b0110 64K 4'b0111~4'b1111 保留
[7:4]	Size	只读	全零。 模式选择位： 1: 片上内存 (local memory 模式) 0: 高速缓存 (cache 模式)
[3:2]	Reserved	只读	复位值：0
[1]	MS	读写	指令 SPM 使能位： 复位值：0
[0]	EN	读写	复位值：0

### 3.2.34. 数据 SPM 寄存器(CP15\_CR10)

此寄存器用以描述数据 SPM 的物理地址基址 (base address)。由于 CK610S 现在只支持一块数据 SPM，所以此寄存器只支持一块内存空间的映射。

31	12	11	8	7	4	3	2	1	0
Base Address		Reserved		Size		Reserve		MS	EN

图表 寄存器描述-37 数据 SPM 寄存器

复位值：0x0000\_00X0  
X 表示 SPM 的大小

[31:12]	Base Address	读写	描述数据 SPM 的基址 (base address)。此地址为物理地址。用户在配置基址时，需要按照 SPM 块的起始地址对齐此基址。例如：配置 8KB 大小的 SPM，基址必须设置成 20'b?????????????????????0。 复位值：0
[11:8]	Reserved	只读	全零。 表示 SPM 大小： 4'b0000 0K 4'b0010 4K (最小) 4'b0011 8K 4'b0100 16K 4'b0101 32K 4'b0110 64K 4'b0111~4'b1111 保留
[7:4]	Size	只读	0
[3:2]	Reserved	只读	模式选择位： 1: 片上内存 (local memory 模式) 0: 高速缓存 (cache 模式)
[1]	MS	读写	复位值：0
[0]	EN	读写	数据 SPM 使能位： 复位值：0

### 3.2.35. DMA 使能寄存器(CP15\_CR11)

DMA 使能寄存器包含一个 OP 字段来控制 DMA 的操作行为。

31		2		1 0
Reserved				OP

图表 寄存器描述-38 DMA 使能寄存器

复位值: 0x0000\_0000

[31:2]	Reserved	只读	全零 用于控制 DMA 的操作行为
[1:0]	OP	读写	00 停止 (STOP) 01 开始 (START) 10, 11 保留

用户在启动 DMA 之前，需要配置完成 DMA 控制寄存器 (CP15\_CR12)，DMA 外部起始地址寄存器 (CP15\_CR14)，DMA 内部起始地址寄存器 (CP15\_CR15) 和 DMA 传输大小寄存器 (CP15\_CR16)。配置 DMA 使能寄存器 (CP15\_CR11) 是控制 DMA 操作的最后一步。

当 DMA 运行时，设置停止 (STOP) 后，DMA 在完成当前传输后中断操作，进入空闲状态，以此表示 DMA 执行被中断。

### 3.2.36. DMA 控制寄存器(CP15\_CR12)

DMA 控制寄存器需要在 DMA 开始传输前完成配置。用户根据应用配置此寄存器，用以描述 DMA 的传输行为。

31	30	29	28	27	4	3	2	1 0
MS	DT	IC	IE	Reserved		FA	Reserved	TS

图表 寄存器描述-39 DMA 控制寄存器

Reset Value:0x0000\_0000

[31]	MS	读写	指令/数据 SPM 选择位: 0: 选择指令 SPM 1: 选择数据 SPM
[30]	DT	读写	传输方向: 0 数据从外部地址传输进入 SPM 内部 1 数据从 SPM 内部传输到外部地址
[29]	IC	读写	传输结束中断使能: 0 当 DMA 传输完成时，不产生 DMA 传输完成中断 1 当 DMA 传输完成时，产生 DMA 传输完成中断
[28]	IE	读写	传输错误中断使能: 0: 传输过程中忽略传输错误，不产生 DMA 传输错误中断 1: 传输过程中，每发生一个传输错误，产生一个中断
[27:3]	Reserved	只读	全零 固定地址模式:
[3]	FA	读写	1: 固定外部地址传输模式，此模式用于和片外 FIFO 进行数据传输 0: 外部地址自动增加模式
[2]	Reserved	只读	全零
[1:0]	TS	读写	传输数据尺寸 00: 字节 01: 半字 10: 字

			指令/数据 SPM 选择位:
[31]	MS	读写	0: 选择指令 SPM
			1: 选择数据 SPM
			11: 未定义

### 3.2.37. DMA 状态寄存器(CP15\_CR13)

DMA 状态寄存器用于表征 DMA 的当前状态。

31	8	7	6	5	4	3	2	1 0
	Reserved		CE	ME	AE	SE	Reserved	ST

图表 寄存器描述-40 DMA 状态寄存器

复位值: 0x0000\_0000

[31:8]	Reserved	只读	全零 DMA 配置异常
[7]	CE	只读	1: 软件设置了 ISPM/DSPM 而硬件上不存在 ISPM/DSPM 时, 发生此异常 0: 无 DMA 配置异常 非对齐异常
[6]	ME	只读	1: 传输过程中发生了非对齐异常 0: 无非对齐异常 访问错误
[5]	AE	只读	1: DMA 发生访问错误异常 0: 无 DMA 访问错误异常 DMA 传输尺寸错误
[4]	SE	只读	1: CP15_CR12 寄存器中的 TS 位被设置成 2'b11(此数值为未定义数值) 0: 无 DMA 传输尺寸错误
[3:2]	Reserved	只读	全零 DMA 当前工作状态: 00: 空闲 (Idle) 01: 繁忙 (Busy) 10: 错误 (Error) 11: 保留 (Reserved)
[1:0]	ST	只读	

### 3.2.38. DMA 外部起始地址寄存器(CP15\_CR14)

DMA 外部起始地址寄存器用以设置传输时的片外内存起始物理地址。用户在设置时需要根据 DMA 控制寄存器中设置的传输尺寸来设置此寄存器, 否则会发生非对齐异常。

31	0
External Start Addr	

图表 寄存器描述-41 DMA 外部起始地址寄存器

复位值: 0x0000\_0000

### 3.2.39. DMA 内部起始地址寄存器(CP15\_CR15)

DMA 内部起始地址寄存器用以设置传输时的片内缓存的起始地址。此起始地址必须落在 SPM 的空间内。在传输时, 只有低位比特有效。例如, 对于 4K 的 ISPM, 只有低 12 位地址有效。高位地址被自动丢弃。

31	0
Internal Start Addr	

图表 寄存器描述-42 DMA 内部起始地址寄存器

复位值:0x0000\_0000

### 3.2.40. DMA 传输大小寄存器(CP15\_CR16)

此寄存器用于配置一次传输过程中，所要传输数据的 byte 大小。当用户采用半字或者字传输方式进行传输时，此传输数据的大小要和传输方式匹配。例如，当进行字传输方式传输时，设置此寄存器的最低 2 bit 位为 0。如果要传输 64KB 的数据，需要设置此寄存器为 0x10000。

DMA 在进行传输时，不会确认所要传输的数据量是否小于 SPM 的实际容量，所以用户需要保证配置此寄存器的正确性。

31	0
Size	

图表 寄存器描述-43 DMA 传输大小寄存器

复位值: 0x0000\_0000

## 4. 指令集

本章讲述 C-SKY CPU 的指令集并给出指令列表，图表 指令集-57 列出了 C-SKY CPU 的指令码。

### 4.1. 指令类型和寻址方式

所有的 C-SKY CPU 指令都是 16 比特长，立即数和偏移量被编码在指令中。其它的操作数采用寄存器寻址，寄存器的内容可以通过存取指令存储到内存或者从内存中取出。

C-SKY CPU 实现了三种指令：

- 跳转指令；
- 内存存取指令；
- 寄存器操作指令。

跳转指令改变指令序列的执行顺序。内存存取指令将通用寄存器的内容存到内存中去，或者从内存中取出数据到通用寄存器中。寄存器操作指令对通用寄存器进行操作，或者访问控制寄存器。指令的格式将在下面的表格中列出。

#### 4.1.1. 寄存器操作指令

以下列出 C-SKY CPU 在寄存器操作指令中所支持的六种寻址方式。

##### 4.1.1.1. 一元寄存器寻址模式

一元寄存器寻址利用指令中一段单独的 4 比特长的寄存器说明区标示操作的源寄存器或目的寄存器。运用这种格式的指令包括 abs, asrc, brev, clrf, clrt, decf, decgt, declt, decne, dect, ffl, incf, inct, ls1c, lsrc, mvc, mvcc, not, sextb, sexth, tstdbz, xsr, zextb, zexth 等。

15	4	3	0
操作码	辅助操作码	寄存器 RX	

图表 指令集-44 一元寄存器寻址格式

##### 4.1.1.2. 二元寄存器寻址模式

二元寄存器寻址在指令中划分两段 4 比特长的寄存器说明区，第一个用于标示源寄存器，第二个用于标示另一源寄存器或目的寄存器。在有些指令中，只有第一个源寄存器值被用到；第二个寄存器说明符仅被用作标示目的寄存器。运用这种格式的指令包括 addc, addu, and, andn, asr, bgenr, cmp[hs,ls,ne], ixh, ixw, ls1, ls1r, mov, movf, movt, mult, or, rsub, subc, subu, tst, xor 等。

15	8	7	4	3	0
操作码	辅助操作码	寄存器 RY	寄存器 RX		

图表 指令集-45 二元寄存器寻址格式

##### 4.1.1.3. 寄存器五位立即数寻址模式

在寄存器五位立即数寻址指令中有一段 4 比特长的寄存器说明区以标示源/目的寄存器，还有一段 5 比特区域用于标示作为第二个源操作数的无符号立即数。运用这种格式的指令包括 andi, asri, bclri, bgeni, bmaski, bseti, btsti, cmpnei, lsli, lsri, rotli, rsubi 等。

15	9	8	4	3	0
操作码	辅助操作码	五位立即数	寄存器 RX		

图表 指令集-46 寄存器五位立即数寻址格式

##### 4.1.1.4. 寄存器五位立即偏移量寻址模式

在寄存器五位立即偏移量寻址指令中有一段 4 比特长的寄存器说明区以标示源/目的寄存

器，还有一段 5 比特区域用于标示作为第二个源操作数的无符号立即数值。这个二进制编码的立即数值偏移加一成为实际立即数值，因此立即偏移量的范围限定在 1 至 32。(五位二进制码表征 0 至 31)运用这种格式的指令包括 addi, subi, cmplti 等。

15	9	8	4	3	0
操作码	辅助操作码	立即偏移量		寄存器 RX	

图表 指令集-47 寄存器五位立即偏移量寻址格式

#### 4.1.1.5. 寄存器七位立即数寻址模式

在寄存器七位立即数寻址指令中有一段 4 比特长的寄存器说明区以标示目的寄存器，还有一段 7 比特区域用于标示作为第二个操作数的无符号立即数。只有 movi 指令用到这种格式。

15	11	10	4	3	0
操作码	七位立即数			寄存器 RX	

图表 指令集-48 寄存器七位立即数寻址格式

#### 4.1.1.6. 控制寄存器寻址模式

在控制寄存器寻址指令中有一段 4 比特长的寄存器说明区，用于标示一个通用源/目的寄存器；还有一段 5 比特区域标示一个控制寄存器。只有 mfcr 和 mtcr 指令用到这种格式。

15	9	8	4	3	0
操作码	控制寄存器			寄存器 RX	

图表 指令集-49 寄存器七位立即数寻址格式

#### 4.1.2. 内存访问指令

C-SKY CPU 在访问内存操作中支持四种寻址方式。

##### 4.1.2.1. 倍乘四位立即数寻址模式

ld 指令和 st 指令用这种寻址模式进行高效的地址计算。在指令中标示为 RX 的通用寄存器的内容被加到 4 位无符号的立即数区域。这个立即数是根据内存通路的大小而左移的，为的是给访问形成有效的地址。寄存器 RZ 在加载时作为目的寄存器，而在存储时作为源寄存器。

15	12	11	8	7	4	3	0
操作码		寄存器 RZ	四位立即数			寄存器 RX	

图表 指令集-50 倍乘四位立即数寻址格式

##### 4.1.2.2. 加载/存储寄存器象限模式

ldq 指令和 stq 指令利用这种模式将一批临近的寄存器，R4 到 R7 的值传进或者传出内存，其中内存地址被通用寄存器 RX 的内容所指向。R4 到 R7 的值将以上升序传入或传出内存。

15	4	3	0
操作码	辅助操作码		寄存器 RX

图表 指令集-51 加载/存储寄存器象限格式

##### 4.1.2.3. 多寄存器载入/存取模式

ldm 指令和 stm 指令利用这种模式将一批临近的寄存器传进或者传出内存，其中内存地址被通用寄存器 R0 的内容所指向。指令中 RX 区域标示了要传值的寄存器列表的第一个寄存器名。RF 到 R15 的值将以上升序传入或传出内存。

15	4	3	0
操作码	辅助操作码		寄存器 RX

图表 指令集-52 多寄存器载入/存取格式

#### 4.1.2.4. 加载相对字模式

lw 指令用这种格式访问一个由程序计数器(PC)相对定位的 32 比特长的字。8 比特转移区域的无符号扩展值再左移 2 位，加上 PC+2 的值就可以得到有效地址。这个值的低两位改为 00 后所定位的 word 被拿到通用寄存器 RZ 中。

15	12	11	8	7	0
操作码		寄存器 RZ		八位转移区	

图表 指令集-53 加载相对字格式

#### 4.1.3. 跳转指令

C-SKY CPU 跳转指令支持三种寻址方式。

##### 4.1.3.1. 位移位偏移方式

br, bf, bt 和 bsr 指令使用这种方式进行目标地址计算。程序记数器加 2 (PC+2) 之后再加上 11 位偏移量经过左移 1 位并且经过有符号扩展后的值。

15	11	10	0
操作码		11 位偏移量	

图表 指令集-54 11 比特移位偏移寻址方式

##### 4.1.3.2. 寄存器寻址方式

jmp 和 jsr 指令使用这种寻址方式进行快速的地址计算。目标地址被存放在由指令 Rx 域指定的寄存器中。

15	4	3	0
指令码		Rx 域	

图表 指令集-55 寄存器寻址方式

##### 4.1.3.3. 间接寻址

jmpi 和 jsri 指令使用这种寻址方式从内存中取出一个 32 比特字并赋给指令记数器，内存地址由当前 PC+2 的值加上一个 8 比特的立即数无符号扩展并左移两位后的值指定。取出的值赋给程序记数器 (PC) 如果这个值是偶数，程序从新的程序记数器指定的位置开始执行，否则将会产生未对齐错误异常。

15	8	7	0
指令码		8 比特偏移量	

图表 指令集-56 间接寻址指令格式

#### 4.2. 指令码表

图表 指令集-57 指令编码表

指令码						指令名称
0	0	0	0	0	0	bkpt
0	0	0	0	0	0	sync
0	0	0	0	0	0	rte
0	0	0	0	0	0	rfi
0	0	0	0	0	0	stop
0	0	0	0	0	0	wait
0	0	0	0	0	0	doze



0	0	0	0	0	0	0	1	1	s	s	s	s	r	r	r	r	mult
0	0	0	0	0	0	1	0	1	s	s	s	s	r	r	r	r	subu
0	0	0	0	0	0	1	1	0	s	s	s	s	r	r	r	r	addc
0	0	0	0	0	0	1	1	1	s	s	s	s	r	r	r	r	subc
0	0	0	0	0	1	0	0	s	s	s	s	s	r	r	r	r	cprgr <sup>cp</sup>
0	0	0	0	0	1	0	1	0	s	s	s	s	r	r	r	r	movf
0	0	0	0	0	1	0	1	1	s	s	s	s	r	r	r	r	lsr
0	0	0	0	0	1	1	0	0	s	s	s	s	r	r	r	r	cmphs
0	0	0	0	0	1	1	0	1	s	s	s	s	r	r	r	r	cmplt
0	0	0	0	0	1	1	1	0	s	s	s	s	r	r	r	r	tst
0	0	0	0	0	1	1	1	1	s	s	s	s	r	r	r	r	cmpne
0	0	0	0	1	0	0	0	c	c	c	c	c	r	r	r	r	mfcr
0	0	0	0	1	0	0	0	1	1	1	1	1	0	b	b	b	psrclr
0	0	0	0	1	0	0	0	1	1	1	1	1	1	b	b	b	psrset
0	0	0	0	1	0	0	1	0	s	s	s	s	r	r	r	r	mov
0	0	0	0	1	0	0	1	1	s	s	s	s	r	r	r	r	bgenr
0	0	0	0	1	0	1	0	0	s	s	s	s	r	r	r	r	rsub
0	0	0	0	1	0	1	0	1	s	s	s	s	r	r	r	r	lxw
0	0	0	0	1	0	1	1	0	s	s	s	s	r	r	r	r	and
0	0	0	0	1	0	1	1	1	s	s	s	s	r	r	r	r	xor
0	0	0	0	1	1	0	0	c	c	c	c	c	r	r	r	r	mtcr
0	0	0	0	1	1	0	1	0	s	s	s	s	r	r	r	r	asr
0	0	0	0	1	1	0	1	1	s	s	s	s	r	r	r	r	lsl
0	0	0	0	1	1	1	0	0	s	s	s	s	r	r	r	r	addu
0	0	0	0	1	1	1	0	1	s	s	s	s	r	r	r	r	lxh
0	0	0	0	1	1	1	1	0	s	s	s	s	r	r	r	r	or
0	0	0	0	1	1	1	1	1	s	s	s	s	r	r	r	r	andn
0	0	1	0	0	0	0	i	i	i	i	i	i	r	r	r	r	addi
0	0	1	0	0	0	1	i	i	i	i	i	i	r	r	r	r	cmplti
0	0	1	0	0	0	1	i	i	i	i	i	i	r	r	r	r	subi
0	0	1	0	0	1	1	s	s	s	s	s	s	r	r	r	r	cpwgr <sup>cp</sup>
0	0	1	0	1	0	0	i	i	i	i	i	i	r	r	r	r	rsubi
0	0	1	0	1	0	1	i	i	i	i	i	i	r	r	r	r	cmpnei
0	0	1	0	1	1	0	0	0	0	0	0	0	r	r	r	r	bmaski #32(set)
0	0	1	0	1	1	0	0	0	0	0	0	1	r	r	r	r	divu
0	0	1	0	1	1	0	0	0	0	0	1	0	r	r	r	r	mflos <sup>610E</sup>
0	0	1	0	1	1	0	0	0	0	0	1	1	r	r	r	r	mfhis <sup>610E</sup>

0 0 1 0	1 1 0 0	0 1 0 0	r r r r	mtlo <sup>620</sup>			
0 0 1 0	1 1 0 0	0 1 0 1	r r r r	mthi <sup>620</sup>			
0 0 1 0	1 1 0 0	0 1 1 0	r r r r	mflo <sup>620</sup>			
0 0 1 0	1 1 0 0	0 1 1 1	r r r r	mfhi <sup>620</sup>			
0 0 1 0	1 1 0 0	1 i i i	r r r r	bmaski			
0 0 1 0	1 1 0 1	i i i i	r r r r	bmaski			
0 0 1 0	1 1 1 i	i i i i	r r r r	andi			
0 0 1 1	0 0 0 i	i i i i	r r r r	bclri			
0 0 1 1	0 0 1 0	0 0 0 0	r r r r	cpwir <sup>cp</sup>			
0 0 1 1	0 0 1 0	0 0 0 1	r r r r	divs			
0 0 1 1	0 0 1 0	0 0 1 0	r r r r	cprsr <sup>cp</sup>			
0 0 1 1	0 0 1 0	0 0 1 1	r r r r	cpwsr <sup>cp</sup>			
0 0 1 1	0 0 1 0	0 1 0 0	r r r r	--			
0 0 1 1	0 0 1 0	0 1 0 1	r r r r	--			
0 0 1 1	0 0 1 0	0 1 1 0	r r r r	--			
0 0 1 1	0 0 1 0	0 1 1 1	r r r r	bgeni			
0 0 1 1	0 0 1 0	1 i i i	r r r r	bgeni			
0 0 1 1	0 0 1 1	i i i i	r r r r	bgeni			
0 0 1 1	0 1 0 i	i i i i	r r r r	bseti			
0 0 1 1	0 1 1 i	i i i i	r r r r	btsti			
0 0 1 1	1 0 0 0	0 0 0 0	r r r r	xsr			
0 0 1 1	1 0 0 i	i i i i	r r r r	rotli			
0 0 1 1	1 0 1 0	0 0 0 0	r r r r	asrc			
0 0 1 1	1 0 1 i	i i i i	r r r r	asri			
0 0 1 1	1 1 0 0	0 0 0 0	r r r r	lslc			
0 0 1 1	1 1 0 i	i i i i	r r r r	lsli			
0 0 1 1	1 1 1 0	0 0 0 0	r r r r	lsrc			
0 0 1 1	1 1 1 i	i i i i	r r r r	lsri			
0 1 0 0	0 0 0 0	s s s s	r r r r	omflip0 <sup>620</sup>			
0 1 0 0	0 0 0 1	s s s s	r r r r	omflip1 <sup>620</sup>			
0 1 0 0	0 0 1 0	s s s s	r r r r	omflip2 <sup>620</sup>			
0 1 0 0	0 0 1 1	s s s s	r r r r	omflip3 <sup>620</sup>			
0 1 0 0	0 1 x x	s s s s	r r r r	--			
0 1 0 0	1 x x x	s s s s	r r r r	--			
0 1 0 1	0 0 0 0	s s s s	r r r r	muls <sup>610E</sup>			
0 1 0 1	0 0 0 1	s s s s	r r r r	mulsa <sup>610E</sup>			

0	1	0	1	0	0	1	0	s	s	s	s	r	r	r	r	mulss <sup>610E</sup>
0	1	0	1	0	0	1	1	s	s	s	s	r	r	r	r	--
0	1	0	1	0	1	0	0	s	s	s	s	r	r	r	r	mulu <sup>610E</sup>
0	1	0	1	0	1	0	1	s	s	s	s	r	r	r	r	mulua <sup>610E</sup>
0	1	0	1	0	1	1	0	s	s	s	s	r	r	r	r	mulus <sup>610E</sup>
0	1	0	1	0	1	1	1	s	s	s	s	r	r	r	r	--
0	1	0	1	1	0	0	0	s	s	s	s	r	r	r	r	vmulsh <sup>610E</sup>
0	1	0	1	1	0	0	1	s	s	s	s	r	r	r	r	vmulsha <sup>610E</sup>
0	1	0	1	1	0	1	0	s	s	s	s	r	r	r	r	vmulshs <sup>610E</sup>
0	1	0	1	1	0	1	1	s	s	s	s	r	r	r	r	--
0	1	0	1	1	1	0	0	s	s	s	s	r	r	r	r	vmulsw <sup>610E</sup>
0	1	0	1	1	1	0	1	s	s	s	s	r	r	r	r	vmulswa <sup>610E</sup>
0	1	0	1	1	1	1	0	s	s	s	s	r	r	r	r	vmulsws <sup>610E</sup>
0	1	0	1	1	1	1	1	s	s	s	s	r	r	r	r	--
0	1	1	0	0	i	i	i	i	i	i	i	r	r	r	r	movi
0	1	1	0	1	0	0	0	s	s	s	s	r	r	r	r	mulsh
0	1	1	0	1	0	0	1	s	s	s	s	r	r	r	r	mulsha <sup>610E</sup>
0	1	1	0	1	0	1	0	s	s	s	s	r	r	r	r	mulshs <sup>610E</sup>
0	1	1	0	1	0	1	1	s	s	s	s	s	r	r	r	cprcr <sup>cp</sup>
0	1	1	0	1	1	0	0	s	s	s	s	r	r	r	r	mulsw <sup>610E</sup>
0	1	1	0	1	1	0	1	s	s	s	s	r	r	r	r	mulswa <sup>610E</sup>
0	1	1	0	1	1	1	0	s	s	s	s	r	r	r	r	mulsws <sup>610E</sup>
0	1	1	0	1	1	1	1	s	s	s	s	s	r	r	r	cpwcr <sup>cp</sup>
0	1	1	1	z	z	z	z	d	d	d	d	d	d	d	d	lrw
0	1	1	1	0	0	0	0	d	d	d	d	d	d	d	d	jmpi
0	1	1	1	1	1	1	1	d	d	d	d	d	d	d	d	jsri
1	0	0	0	z	z	z	z	i	i	i	i	r	r	r	r	ld.w
1	0	0	1	z	z	z	z	i	i	i	i	r	r	r	r	st.w
1	0	1	0	z	z	z	z	i	i	i	i	r	r	r	r	ld.b
1	0	1	1	z	z	z	z	i	i	i	i	r	r	r	r	st.b
1	1	0	0	z	z	z	z	i	i	i	i	r	r	r	r	ld.h
1	1	0	1	z	z	z	z	i	i	i	i	r	r	r	r	st.h
1	1	1	0	0	d	d	d	d	d	d	d	d	d	d	d	bt
1	1	1	0	1	d	d	d	d	d	d	d	d	d	d	d	bf
1	1	1	1	0	d	d	d	d	d	d	d	d	d	d	d	br
1	1	1	1	1	1	d	d	d	d	d	d	d	d	d	d	bsr

参数说明:

参数	定义	参数	定义
----	----	----	----

rrrr	RX 域	d..d	跳转指令偏移量
ssss	RY 域	b..b	寄存器位索引
zzzz	RZ 域	x..x	未定义
ffff	RF 域	--	保留
ccccc	控制寄存器索引	i..i	立即数域

图表 指令集-58 指令编码表参数说明

**注意:** bgeni、bseti 等指令具有多种指令码

若使用 GNU 编程套件，更多编程细节，请参阅：<GNU Assembler>, <GNU C Compiler>, <GNU Linker>, <GNU objcopy>, <GNU objdump>, <GNU make>。

## 5. 指令流水线

本章介绍关于 C-SKY CPU 的指令流水线和指令时序信息。

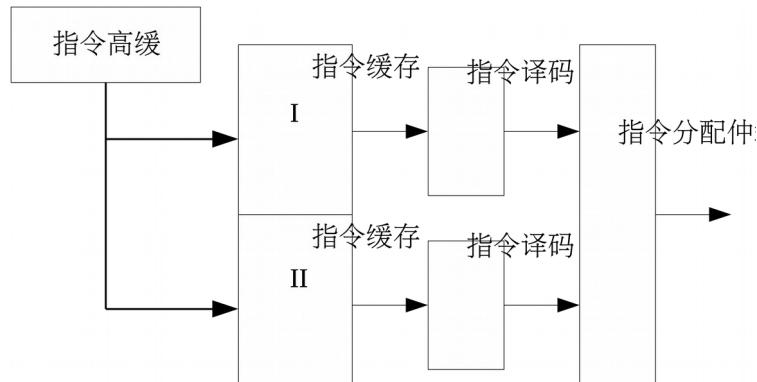
C-SKY CPU 微处理器有 8 级流水线：即指令读取 I、指令读取 II、指令译码、访问寄存器组、指令执行、访问数据高缓 I、访问数据高缓 II、数据回写等 8 级。该处理器的指令高缓和数据高缓都是可配置的（2K/4K/8K/16K/32K/64K）。8 级流水线的作用如图表 指令流水线-59：

流水线名称	缩写	流水线作用
指令读取 I	IF	1、访问指令高缓； 2、计算下一条指令的地址； 3、访问跳转历史单元，分支预测。
指令读取 II	IS	1、选取将要执行指令； 2、指令预译码； 3、双发指令仲裁分配。
指令译码	ID	1、指令译码； 2、指令相关性分析。
访问寄存器组	RF	1、从寄存器组中读取源操作数； 2、指令译码并检查指令互锁条件； 3、指令发射到执行单元。
指令执行	EX	1、大多数 ALU 指令在该级完成； 2、Load/Store 指令的数据地址在该级产生； 3、跳转指令在该级检查跳转条件并产生跳转目的地址。
访问数据高缓 I	DF	访问数据高缓。
访问数据高缓 II	DS	数据对齐并完成 Load/Store 指令。
数据回写	WB	指令执行结果回写到寄存器组。

图表 指令流水线-59 各级流水线作用

在流水执行指令的过程中，采用双发射技术，即在条件许可的情况下，一个时钟周期发射两条指令；同时采用非阻塞发射架构，当指令之间有相关性的时候，有相关性的指令会被发射到各个执行单元的保留栈中，将后续指令中没有相关性的指令直接发射到执行单元先执行；采用寄存器重命名技术，消除部分伪相关的指令相关性；另外数据前馈硬件将前一条指令执行结果直接反馈到执行单元作为它的源操作数，从而使得后面的指令不用等到前一条指应回写后才得到操作数。

C-SKY CPU 包含一个 32 位指令缓存用于给 C-SKY CPU 指令寄存器提供指令。指令提取单元每次从指令高缓中取 32 位数据写入指令缓存，即每次向缓存中写入两条指令。当执行指令时，两条指令同时送给指令译码单元用于译码。指令分配仲裁根据两条指令的译码结果决定，能否将两条指令同时传向后级流水线。当两条指令由于出现结构性冲突不能同时传向后级流水线时，对于当前两条指令的分配将分别在两个时钟周期按序逐条完成。当指令缓存内的指令都执行完成之后，指令提取单元重新从指令高速缓存中取出两条指令，填充指令缓存。图表 指令流水线-60 显示了指令执行时的数据流向。



图表 指令流水线-60 指令执行时的数据流向

单周期指令流水线重叠执行顺序如图表 指令流水线-61 所示，这类指令一般都是按顺序发射并完成。大多数的算术和逻辑指令都属于这类指令。

IF	IS	ID	RF	EX	WB
----	----	----	----	----	----

IF	IS	ID	RF	EX	WB	
IF	IS	ID	RF	EX	WB	
IF	IS	ID	RF	EX	WB	
IF	IS	ID	RF	EX	WB	

图表 指令流水线-61 单周期指令流水线重叠执行

load, store, 乘法指令, 除法指令, Omflip 指令至少需要两个执行周期才能完成。load, store 指令在 EX 级计算目标数据地址, 在 DF 级访问数据缓存。其执行过程如图表 指令流水线-62:

时间刻度



图表 指令流水线-62 LD/ST 指令执行过程

乘法指令需要两个执行周期完成（可流水操作），如图表 指令流水线-63 所示：

时间刻度



图表 指令流水线-63 乘法指令执行过程

除法指令需要 1-32 个执行周期完成，如图表 指令流水线-64 所示：

时间刻度



图表 指令流水线-64 除法指令执行过程

br, bsr 指令的跳转和 bt, bf 指令预测跳转且预测正确时，流水线不停顿，其执行过程如图表 指令流水线-65 所示：

时间刻度



图表 指令流水线-65 BR, BSR 指令的跳转和 BT, BF 指令预测跳转且预测正确时执行过程

bt, bf 指令预测不跳转且预测正确时，流水线不停顿，其执行过程如图表 5-8 所示：

时间刻度



图表 指令流水线-66 BT, BF 指令预测不跳转且预测正确时执行过程

**jmp** 指令和 **bt**、**bf** 指令预测不正确时的跳转需要 5 个周期来填充流水线，其执行过程如图表 指令流水线-67 所示：

时间刻度



图表 指令流水线-67 JMP, BT 和 BF 指令预测不正确时的执行过程

对于 **jmp r15** 指令，CK610 采用返回栈加速该函数返回指令的执行，当取指单元从返回栈获取 **jmp r15** 指令跳转目标地址并预取正确时，该跳转指令仅需两个周期填充流水线，若预取错误则需要 5 个周期填充流水线。

**jsri** 指令的跳转需要至少 3 个周期来填充流水线，其执行过程如图表 指令流水线-68 所示：

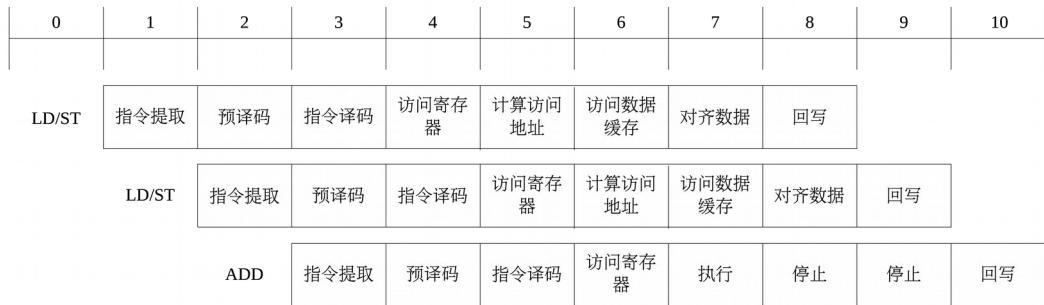
时间刻度



图表 指令流水线-68 JSRI 指令的执行过程

有一些多周期的指令可以流水执行，从而使得每条指令的有效执行时间小于这些指令执行时间的和。这种执行方法的限制是指令和指令之间不能有数据的相关性，并且指令必须按顺序结束和回写。当多周期指令后面有一条单周期指令时，该单周期指令必须等到前一条指令结束之后才能结束，以满足指令按序执行的需要。图表 指令流水线-69 显示了两条没有数据相关性的 LD/ST 指令后跟一条单周期指令 ADD 的执行过程：

时间刻度



图表 指令流水线-69 简单的指令流水线执行过程

对于访问存储区的指令，可能有等待状态。这会导致所有在访问存储区指令之后的指令处于停止状态，因为必须等到访问存储区的指令完成之后这些指令才能完成，其执行过程如图表 指令流水线-70 所示：

时间刻度



图表 指令流水线-70 带有等待状态的指令流水执行过程

lrw 指令需要 3 个周期才能完成，其目标地址计算在执行级完成。图表 指令流水线-71 显示了执行两条 lrw 指令和一条没有数据相关性的访问存储区指令的执行过程。

时间刻度



图表 指令流水线-71 LRW 指令与访问存储指令混合的执行过程

## 6. 异常处理

异常处理(包括指令异常和外部中断)是处理器的一项重要技术，在某些异常事件产生时，用来使处理器转入对这些事件的处理。这些事件包括硬件错误，指令执行错误，和用户请求服务等等。本章主要描述异常种类、异常优先级、异常向量表、异常返回和总线错误恢复等内容。

### 6.1. 异常处理概述

异常处理是处理器根据内部或外部的异常事件从正常的程序处理转入特定的异常处理程序。引起异常的外部事件包括：外部设备的中断请求、硬断点请求、读写访问错误和硬件重启；引起异常的内部事件包括：非法指令、不对齐错误（misaligned error）、特权异常和指令跟踪，trap 和 bkpt 指令正常执行时也会产生异常。而且，非法指令、load 和 store 访问的地址没有对齐、jmpi 和 jsri 跳转到奇地址还有用户模式下执行特权指令都会产生异常。异常处理利用异常向量表和一组影子寄存器跳转到异常服务程序的入口。

异常处理的关键就是在异常发生时，保存 CPU 当前指令运行的状态，在退出异常处理时恢复异常处理前的状态。异常能够在指令流水线的各个阶段被识别，并使后面的指令不会改变 CPU 的状态。异常在指令的边界上被处理，即 CPU 在指令退休时响应中断，并保存退出异常处理时下一条被执行的指令的地址。即使异常在 IS 阶段（非法指令、jmpi 和 jsri 的目标地址是奇、断点异常、访问错误）或 RF 阶段（trap 和 bkpt）被识别，异常也要在相应的指令退休时才会被处理。为了异常处理不影响 CPU 的性能，CPU 在异常处理结束后要避免重复执行以前的指令。C-SKY CPU 根据异常识别时的指令是否完成决定异常地址寄存器存储哪一条指令的地址。例如，如果异常事件是外部中断服务请求，被中断的指令将正常退休并改变 CPU 的状态，它的下一条指令的地址将被保存在异常地址寄存器中作为中断返回时指令的入口；如果异常事件是由除以零的除法指令产生的，因为这条指令不能完成，它将异常退休但不改变 CPU 的状态(即不改变寄存器的值)，这条除法指令的地址将被保存在异常地址寄存器中，CPU 从中断服务程序返回时继续执行这条除法指令。

异常按以下步骤被处理：

第一步，处理器更新 PSR 中的 VEC，然后保存 PSR 和 PC 到影子寄存器中。对于快速中断，PSR 和 PC 被保存到 FPSR 和 FPC 中；对于其它的异常，它们被保存到 EPSR 和 EPC 中。如果 PSR 的 EE 位被清零了，异常事件（除了普通中断和快速中断）将导致不可恢复的错误异常。PSR 和 PC 被保存后，PSR 的 EE 位被清零，处理器就只能处理不可恢复的错误异常。不可恢复的错误异常发生时，EPSR 和 EPC 也会被更新。

第二步，处理器设置 PSR 的 S 位进入超级用户模式，并且把 PSR 的 TM 位清零防止异常服务程序被跟踪。中断使能位 PSR 的 IE 位也被清零禁止响应中断。如果异常是快速中断或重启，快速中断使能 PSR 的 FE 位会被清零，但是其它的异常不影响 PSR 的 FE 位。

传输控制位 PSR 的 TE 位被清零防止外部存储器管理单元对在异常入口地址计算时进行地址转译，使下面的访问以非转译的方式进行。

处理器还要决定对应的异常向量。对于向量中断，异常向量由外部的中断控制器提供；对于其它的异常，处理器根据内部逻辑决定异常向量。异常向量用来计算异常服务程序的入口地址，并被保存在 PSR 的 VEC 中以支持共享异常服务的情况。

最后一步，处理器计算异常服务程序的第一条指令的地址并将 CPU 的控制权转交给异常服务程序。处理器把异常向量乘以 4 再加上异常向量基准地址（存在向量基准地址寄存器中）就得到了异常向量表中对应的异常入口地址。处理器用这个入口地址从存储器中读取一个字，这个异常入口地址的[31:1]转载到程序计数器中作为异常服务程序的第一条指令的地址（PC 的最低位始终是 0，与异常向量表中取得的异常入口地址值的最低位无关）。同时，处理器把这个异常入口地址的最低位装载到 PSR (AF) 位，它用来控制异常处理程序使用哪一个寄存器组。就这样，处理器开始执行新的指令。

所有的异常向量存放在超级用户地址空间，并以指令空间索引（TC1=1）访问。在处

理器地址映射中，只有重启向量是固定的。一旦处理器完成初始化，VBR 允许异常向量表的基准地址被重载。

C-SKY CPU 支持 512 个字节的向量表包含 128 个异常向量（见图表 异常处理-72）。开始的 31 个向量是用作在处理器内部识别的向量。第 32 个向量是留给软件的，用作指向系统描述符的指针。其余的 96 个向量是留给外部设备的。外部设备通过 7 位的中断向量和中断请求使处理器响应中断服务。处理器响应中断请求时锁存这个中断向量。对那些不能提供中断向量的设备，处理器为一般中断和快速中断提供了自动向量。

图表 异常处理-72 异常向量分配

向量号	向量偏移 (十六进制)	向量分配
0	000	重启异常。
1	004	未对齐访问异常。
2	008	访问错误异常。
3	00C	除以零异常。
4	010	非法指令异常。
5	014	特权违反异常。
6	018	跟踪异常。
7	01C	断点异常。
8	020	不可恢复错误异常。
9	024	Idly4 异常。
10	028	普通中断。
11	02C	快速中断。
12	030	保留 (HAI)。
13	034	TLB 不可恢复异常 <sup>610M</sup> 。
14	038	TLB 失配异常 <sup>610M</sup> 。
15	03C	TLB 修改异常 <sup>610M</sup> 。
16—19	040—04C	陷阱指令异常 (TRAP #0-3)。
20	050	TLB 读无效异常 <sup>610M</sup> 。
21	054	TLB 写无效异常 <sup>610M</sup> 。
20—29	058—074	保留。
30	078	协处理器中断和异常 <sup>CP</sup>
31	07C	系统描述符指针。
32—127	080—1FC	保留在向量中断控制器使用。

## 6.2. 异常类型

本节描述外部中断异常和在 C-SKY CPU 内部产生的异常。C-SKY CPU 处理的异常有以下几类：

- 重启异常；
- 未对齐访问异常；
- 访问错误异常；
- 除以零异常；
- 非法指令异常；
- 特权违反异常；
- 跟踪异常；
- 断点异常；
- 不可恢复错误异常/TLB 不可恢复异常<sup>610M</sup>；
- Idly4 异常；
- 普通中断；
- 快速中断；
- TLB 失配异常<sup>610M</sup>；
- TLB 修改异常<sup>610M</sup>；
- 陷阱指令异常；
- TLB 读无效异常<sup>610M</sup>；

- TLB 写无效异常<sup>610M</sup>；
- 协处理器中断和异常<sup>CP</sup>。

### 6.2.1. 重启异常（向量偏移 0X0）

重启异常是所有异常中优先级最高的，它是用于系统初始化和发生重大故障后恢复系统。重启会中止处理器的所有操作，被中止的操作是不可恢复的。重启也在测试时用于初始化扫描链和时钟控制逻辑中锁存器的值，它也同时对处理器进行上电初始化。

重启异常设置 PSR (S) 为高电平使处理器工作在超级用户模式，并且把 PSR (TM) 清零禁止跟踪异常。重启异常也会把 PSR (IE) 和 PSR (FE) 清零以禁止中断响应。同时，VBR (向量基准寄存器) 也被清零，异常向量表的基准地址就是 0X00000000，CPU 从异常向量表中以偏移地址 0X0 为偏移地址读取异常向量，并把它装载到程序计数器 (PC)。异常处理器把控制权转移到 PC 指向的地址。

### 6.2.2. 未对齐访问异常（向量偏移 0X4）

处理器试图在与访问大小不一致的地址边界上执行访问操作，就会发生地址未对齐访问异常。通过设置 PSR (MM)，这个异常可以被屏蔽，处理器会忽略对数据的对齐检查，而访问小于这个未对齐地址又最接近它的地址边界上。EPC 指向试图进行未对齐访问的指令。

未对齐访问异常也可能发生在数据上。如果 jmpi 或 jsri 跳转到奇地址也会引起未对齐访问异常。这种情况下，处理器把 jmpi 或 jsri 跳转的目标地址存在 EPC 中，而不是存 jmpi 或 jsri 的地址。这也是 EPC 中的值为奇数的唯一情况。

**注意：**在这种情况下，如果跟踪模式被使能了，EPSR 中的 TP 不会有效，因而未对齐的 jmpi 和 jsri 不会被跟踪。PSR(MM)也不会屏蔽 jmpi 和 jsri 相关的未对齐访问异常。

### 6.2.3. 访问错误异常（向量偏移 0X8）

如果 pad\_biu\_hresp[1:0]=1，就意味着发生了访问错误异常。

总线上的错误都会引起访问错误异常，使处理器进行异常处理。

当 TLB 访问错误时，外部存储器管理单元也能用访问错误异常来产生异常事件。TLB 缺页异常是用来处理 TLB 缺页的情况。

### 6.2.4. 除以零异常（向量偏移 0X0C）

当处理器发现除法指令的除数是零时，处理器进行异常处理而不执行该除法指令。EPC 指向该除法指令。

### 6.2.5. 非法指令异常（向量偏移 0X10）

处理器译码时如果发现了非法指令或没有实现的指令，该指令不会被执行而进行异常处理。EPC 指向该非法指令。

### 6.2.6. 特权违反异常（向量偏移 0X14）

为了保护系统安全，一些指令被授予了特权，它们只能在超级用户模式下被执行。试图在用户模式下执行下面的特权指令都会产生特权违反异常：MFCR、MTCR、PSRSET、PSRCLR、RFI、RTE、STOP、WAIT、DOZE。

处理器如果发现了特权违反异常，在执行该指令前进行异常处理。EPC 指向该特权指令。

### 6.2.7. 跟踪异常（向量偏移 0X18）

为了便于程序开发调试，C-SKY CPU 提高了对每条指令或对改变控制流指令的跟踪能

力。在指令跟踪模式下，每条指令在执行完后都会产生一个跟踪异常，以便于调试程序监测程序的执行。在跳转跟踪模式下，每条改变控制流的指令（branch, jmp, 等等）都会产生一个跟踪异常。对于条件跳转指令，不管程序有没有跳转，跟踪异常都会发生。

如果处理器工作在跳转跟踪模式下，以下指令都会引起跟踪异常：jmp, jsr, jmpi, jsri, br, bt, bf, bsr。无论条件跳转指令的执行结果如何，跟踪异常都被使能了。

如果被跟踪的指令由于发生了其它的异常而没有完成，跟踪异常不会被处理或被标记为等待处理。被跟踪的 jmpi 和 jsri 指令的目标地址没有对齐也同样处理。如果在指令完成时外部中断被探测到了，跟踪异常将延时处理，在 EPSR 中被标记为等待处理的跟踪异常。

PSR 的 TM 位控制跟踪模式。TM 的状态决定了指令退休时是否产生跟踪异常。请参考第三章 PSR 的 TM 位的定义。

跟踪异常处理起始于被跟踪的指令退休之后且在下一条指令退休之前。EPC 指向下一条指令。

以下控制相关的指令不能被跟踪：rte, rfi, trap, stop, wait, doze 和 bkpt。如果 EPSR (TP) 有效，跟踪异常作为 rte 或 rfi 正常执行的一部分被处理，而不管 PSR 或 EPSR 的 TM 位的状态。

如果在被跟踪的指令退休时处理器发现有中断等待处理，中断的优先级比跟踪异常高，PSR 的影子寄存器的 TP (等待处理的跟踪异常) 将有效，处理器响应中断。中断服务程序的 rte 或 rfi 退休时，等待处理的跟踪异常会被处理。等待处理的跟踪异常是用来跟踪前面的指令的，这条指令已经退休，为了避免这个异常被丢失，它的优先级是最高的仅次于重启。

### 6.2.8. 断点异常 (向量偏移 0X1C)

C-SKY CPU 提供了断点指令 bkpt 和硬断点请求管脚 (pad\_biu\_brkrq\_b[1:0])，它们被分配同一个断点异常向量。请参考第七章接口操作中 pad\_biu\_brkrq\_b[1:0]管脚的操作。为了尽量避免硬断点异常被丢掉，硬断点异常被赋予了比中断更高的优先级。如果 pad\_biu\_brkrq\_b[1:0]设置在指令访问上，相应的指令不会被执行，当它退休时，处理器响应硬断点异常。如果 pad\_biu\_brkrq\_b[1:0]设置在数据访问上，相应的指令被允许完成，当它退休时，硬断点异常被响应。如果断点异常是由于 bkpt 指令或指令访问时 pad\_biu\_brkrq\_b[1:0]设置，EPC 指向该指令。如果数据访问时 pad\_biu\_brkrq\_b[1:0]设置，EPC 指向它的下一条指令。

### 6.2.9. 不可恢复错误异常 (向量偏移 0X20)

当 PSR (EE) 为零时，除了快速中断外的异常会产生不可恢复的异常，因为这时用于异常恢复的信息（存于 EPC 和 EPSR）由于不可恢复的错误而被重写了。

由于所写的软件在 PSR (EE) 为零时应该排除了异常事件发生的可能，这种错误一般意味着有系统错误。在不可恢复错误异常的服务程序中，引起不可恢复错误异常的异常类型是不确定的。

#### 6.2.9.1. TLB 不可恢复异常 610M

在物理地址转换过程中，如果发现多个匹配的表项存在，将发生 TLB 不可恢复异常。

### 6.2.10. IDLY4 异常 (异常偏移 0X24)

idly4 异常用来指示在 idly4 指令序列中发生了传输错误。在该异常服务程序中，EPC 指向引起传输错误的指令。异常服务程序应该分析发生传输错误的原因，并备份 EPC 的值以便在必要时重新执行 idly4 指令序列。

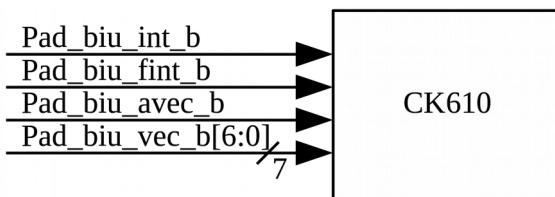
## 6.2.11. 中断异常

当外部设备需要向处理器请求服务或发送处理器需要的信息时，它可以用中断请求信号和相应的中断向量信号向处理器请求中断异常。

尽管处理器可以通过允许多周期指令被中断的方式来缩短中断响应的延时，一般中断在指令的边界上被确认。如果 PSR (IC) 位设置了，divs, divu, ldm, stm, 和 stq 可以被中断而不等它们完成。

C-SKY CPU 提供了两个中断请求信号，支持自动提供中断向量号和由外设显式地提供中断向量号的方式。

图表 异常处理-73 显示了和中断相关的到处理器核的接口信号。为了支持向量化的中断，外设在中断请求时，用 7 位的中断向量信号提供中断向量号，或设置 pad\_biu\_avec\_b 使用预先定义好的中断向量号。如果 pad\_biu\_avec\_b 有效，pad\_biu\_vec\_b[6:0] 上的输入信号被忽略。pad\_biu\_int\_b, pad\_biu\_fint\_b 和 pad\_biu\_avec\_b 都是低电平有效。



图表 异常处理-73 中断接口信号

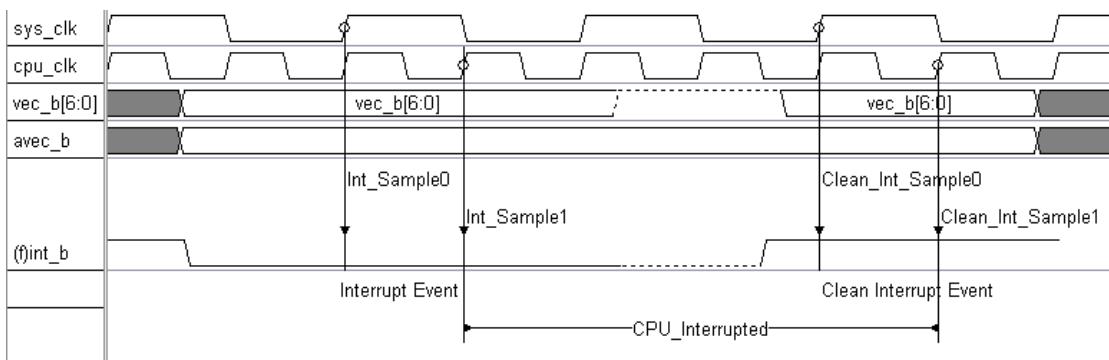
### 6.2.11.1. 普通中断 (INT)

pad\_biu\_int\_b 是普通中断请求引脚。它是中断中优先级最低的。如果 PSR (IE) 被清零了，pad\_biu\_int\_b 输入信号被屏蔽，处理器不响应异常。普通中断使用 EPSR 和 EPC 这一组异常影子寄存器，它也可以被 PSR (EE) 屏蔽。当 pad\_biu\_int\_b 有效时，如果 pad\_biu\_avec\_b 也有效，处理器使用自动向量号，它的向量偏移是 0X28，否则，处理器使用 pad\_biu\_vec\_b[6:0] 上提供的向量号，它可以是 32–127 中的任意一个（硬件上不排除使用 0–31）。

### 6.2.11.2. 快速中断 (FINT)

pad\_biu\_fint\_b 是快速中断请求输入引脚。如果快速中断被使能 (PSR (FE) 有效)，它的优先级比普通中断请求高。快速中断使用 FPSR 和 FPC 这一组异常影子寄存器，它不会被 PSR (EE) 屏蔽。如果 PSR (FE) 清零了，快速中断就被屏蔽了。当 pad\_biu\_fint\_b 有效时，如果 pad\_biu\_avec\_b 也有效，处理器使用自动向量号，它的向量偏移是 0X2C，否则，处理器使用 pad\_biu\_vec\_b[6:0] 上提供的向量号，它可以是 32–127 中的任意一个（硬件上不排除使用 0–31）。

### 6.2.11.3. 中断处理过程



图表 异常处理-74 中断处理过程

在上图中，当中断向量号 pad\_biu\_vec\_b[6:0]和自动中断向量 pad\_biu\_avec\_b 都已经准备好时，拉低普通中断信号线 pad\_biu\_int\_b，该信号经系统时钟 sys\_clk 和 CPU 内部时钟 cpu\_clk 的上升沿依次采样后，CPU 内部收到中断，并根据 pad\_biu\_avec\_b 的设置取得中断向量，进入中断服务程序。在中断服务程序中，应该清除外部中断，即拉高 pad\_biu\_int\_b，此信号亦需经此两个时钟依次采样后才会退出中断。快速中断(fint)的操作亦是如此。

### 6.2.12. TLB 不可恢复异常<sup>610M</sup> (向量偏移 0X34)

在 CPU 读写数据时，如果 MMU 的 jTLB 和 vTLB 表项中的多个表项与虚拟地址匹配时，将发生 TLB 不可恢复异常；在开启带有硬件回填功能的 CPU 时，MMU 访问总线，总线反馈访问错误（access error）时，也会发生 TLB 不可恢复异常。

### 6.2.13. TLB 失配异常<sup>610M</sup> (向量偏移 0X38)

在 CPU 取指或数据访问时，如果 jTLB 中没有与虚拟地址匹配的表项，将发生 TLB 失配异常，软件需要负责再填充 jTLB。

### 6.2.14. TLB 修改异常<sup>610M</sup> (向量偏移 0X3C)

在 CPU 写数据时，如果 MMU 的 jTLB 表项与虚拟地址匹配，但是匹配的表项关闭了页面 dirty 位时发生 TLB 修改异常。

### 6.2.15. 陷阱指令异常 (向量偏移 0X40—0X4C)

一些指令可以用来显示地产生陷阱异常。trap #n 指令可以强制产生异常，它可以用于用户程序的系统调用。在异常服务程序中，EPC 指向 trap 指令。

### 6.2.16. TLB 读无效异常<sup>610M</sup> (向量偏移 0X50)

在 CPU 取指或读数据时，如果 MMU 的 jTLB 表项与虚拟地址匹配，但是匹配的表项关闭了页面有效位时发生 TLB 读无效异常。

### 6.2.17. TLB 写无效异常<sup>610M</sup> (向量偏移 0X54)

在 CPU 写数据时，如果 MMU 的 jTLB 表项与虚拟地址匹配，但是匹配的表项关闭了页面有效位时发生 TLB 写无效异常。

### 6.2.18. 协处理器中断和异常<sup>CP</sup> (向量偏移 0X78)

为了提供对于某些特殊应用背景的加速（如浮点运算的加速），C-SKY CPU 提供了支持最多为 15 个协处理器的协处理器接口，这些协处理器可以有中天公司提供 IP，也可以由用户自行设计协处理器，最后集成到 CPU 中。协处理器中断和异常提供了对于各个协处理器可能出现中断或者异常的支持。

当出现协处理器异常时，执行该指令的协处理器会进入异常态，直到下一条该类型的协处理器指令进入该协处理器，返回协处理器异常。例如当一条浮点指令出现异常后，浮点协处理器会进入异常态，当下一条浮点指令进入浮点协处理器后，将会返回浮点指令异常。注意在此过程中，其他协处理器仍然能够正常工作。

当出现协处理器中断时，按照协处理器窗口寄存器设置的优先级，和其他中断类似的行为执行。

## 6.3. 异常优先级

如图表 异常处理-75 所示，根据异常的特性和被处理的先后关系，C-SKY CPU 把优先级分为 10 级。在图表 异常处理-75 中，1 代表最高优先级，10 代表了最低优先级。值得注意

的是，在第9组中，几个异常共享一个优先级，因为它们之间相互有排斥性。

在C-SKY CPU里，多个异常可以同时发生。重启异常是很特别的，它有最高的优先级。

所有其它的异常按图表 异常处理-75 中的优先级关系进行处理。

如果PSR(EE)被清零了，当异常发生时，处理器处理的是不可恢复异常。

如果多个异常同时发生，拥有最高优先级的异常最先被处理。处理器在异常返回后，重新执行产生异常的指令时，其余的异常可以重现。

图表 异常处理-75 异常优先级

优先级	异常与它相关的优先级	特征
1	重启异常	处理器中止所有程序运行，初始化系统。
2	待处理的跟踪异常	如果TP=1，在 rte 或 rfi 指令退休后，处理器处理待处理的跟踪异常。
3	硬断点请求	在相关的指令退休后，硬断点请求被处理。
4	Id1y4 错误	在相关的指令退休后，处理器保存上下文并处理异常。
5	不对齐错误	在相关的指令退休后，处理器保存上下文并处理异常。
6	协处理器中断和异常	出现协处理器中断时，按照处理中断的方式执行；出现协处理器异常时，按照协处理器处理异常的方式执行。
7.0	快速中断	如果IC=0，中断在指令退休后被响应；如果IC=1，处理器允许中断在指令完成之前就被响应。
7.1	普通中断	
8.0	不可恢复错误异常/TLB 不可恢复异常 <sub>610M</sub>	在相关的指令退休后，处理器保存上下文并处理异常。
8.1	TLB 读无效异常 <sub>610M</sub> /TLB 写无效异常 <sub>610M</sub>	
8.2		
8.3		
8.4	TLB 修改异常 <sub>610M</sub> TLB 失配异常 <sub>610M</sub> 访问错误	
9	非法指令 特权异常 禁止硬件加速器 除以零 陷阱指令 断点指令	在相关的指令退休后，处理器保存上下文并处理异常。
10	跟踪异常	在相关的指令退休后，处理器保存上下文并处理异常。

### 6.3.1. 发生待处理的异常时调试请求

处理器如果在异常发生的同时接收到了调试请求信号，先进入调试模式。异常延后处理，直到处理器退出调试模式和产生异常的指令重新被执行。

## 6.4. 异常返回

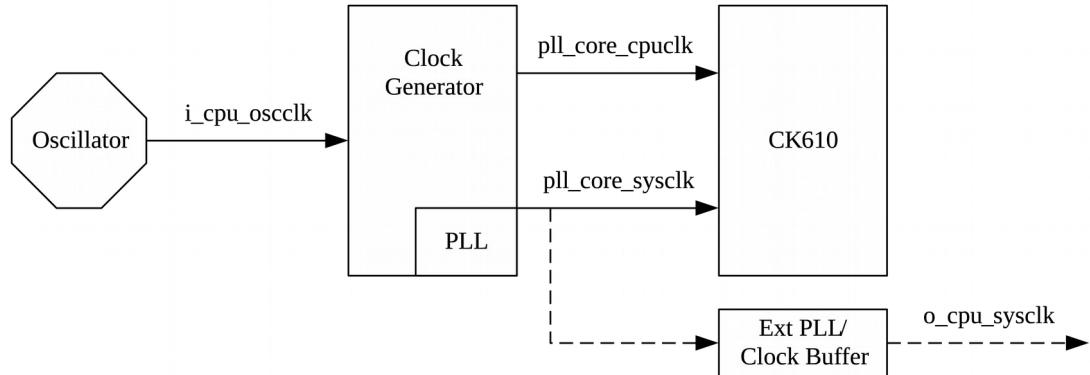
根据正在处理的异常类型，处理器通过执行 rfi 或 rte 指令从异常服务程序中返回。rfi 指令利用保存在 FPSR 和 FPC 影子寄存器中的上下文从异常服务程序中返回； rte 指令利用保存在 EPSR 和 EPC 影子寄存器中的上下文从异常服务程序中返回。

## 7. 接口信号

本章介绍 C-SKY CPU 处理器接口信号，并详细说明信号的特性。其中主要包括信号名，信号类型（输入、输出、双向输入输出等），并简要描述信号的功能。

### 7.1. 时钟信号

C-SKY CPU 采用输出源时钟的方式。与 C-SKY CPU 接口的外部 IP 应采用 C-SKY CPU 提供的源时钟来取得与 C-SKY CPU 的同步。图表 接口信号-76 显示了一个使用 C-SKY CPU 的时钟管理方式。



图表 接口信号-76 C-SKY CPU 的时钟管理方式

全局的系统时钟信号 (`i_cpu_oscclk`) 是从板上输入的系统时钟，为处理器的内部逻辑提供时钟信号。C-SKY CPU 处理器是基于静态操作设计，所以该时钟信号可以设计为门控时钟。相关模块的时序必须是一致的。

为了降低功耗 C-SKY CPU 中广泛的使用了 clock gating 技术，顶层模块和底层子模块均采用了门控。局部的 clock gating 通过 clock gating 单元实现，全局的 clock gating 在 cp0 单元的控制之下。外部晶振产生的时钟由 PLL(锁相环)锁相倍频后，经过选择输出 `pll_core_cpubclk` 和 `pll_core_sysclk` 两个时钟，这两个时钟经过 gated 单元之后得到 cpu 内部各个模块使用的时钟信号。

### 7.2. 接口信号

本节中将解释，可能和其他系统模块和包进行集成的所有输入、输出和双向输入输出端口。

#### 7.2.1. 信号类型

为了易于识别记忆，C-SKY CPU 接口信号使用了图表 7-2 的命名规则：

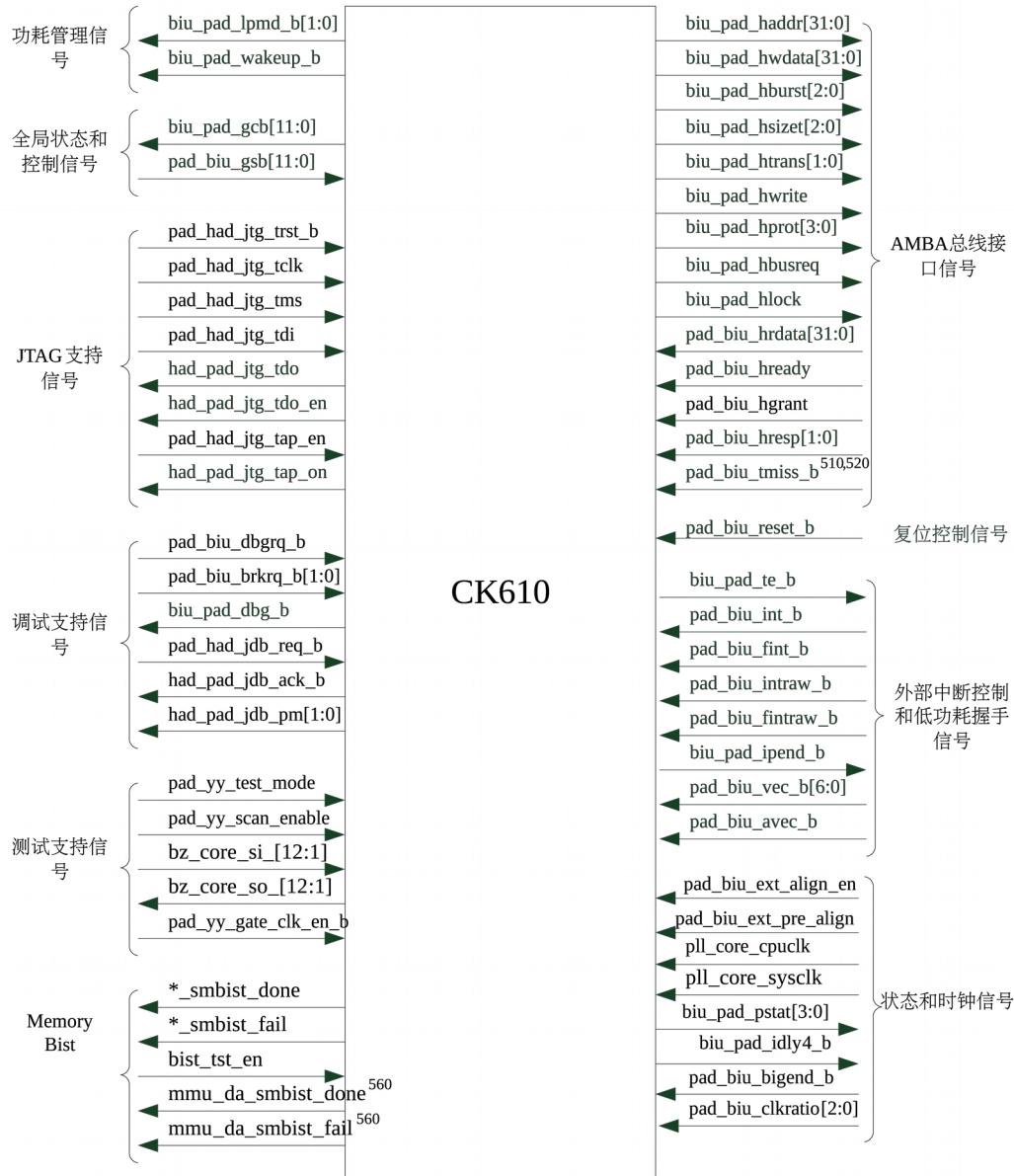
符号	含义
*	多组信号。
bist	BIST(Build-in Self Test)相关信号。
biu	BIU(Bus Interface Unit)相关信号。
core	CORE 相关信号。
had	HAD(Hardware Assist Debug)相关信号。
jdb	JTAG(Joint Test Action Group)硬件协同调试相关信号。
jtg	JTAG(Joint Test Action Group)相关信号。
mmu	mmu 相关信号。
pad	PAD 相关信号。
pll	PLL(Phase-Locked Loop)相关信号。
smbist	SMBIST(Simple Memory Build-in Self Test)相关信号。
xx	多个目标单元的信号。
yy	跨越多个流水线级。

图表 接口信号-77 C-SKY CPU 接口信号命名规则

注意：\_b 后缀用于表示低电平有效，没有\_b 后缀的信号高电平有效。

## 7.2.2. 信号功能

图表 接口信号-78 根据功能对 core 的总线和控制信号进行了分类。



图表 接口信号-78 C-SKY CPU 接口信号

图表 接口信号-79 列出了 C-SKY CPU 各个信号的功能、类型，信号定义和 reset 之后的值。

图表 接口信号-79 C-SKY CPU 各个信号的功能、类型、定义

信号名	I/O	Reset	定义
<b>AMBA 总线接口信号:</b>			
biu_pad_haddr [31:0]	0	-	地址总线: 32 位地址总线。
biu_pad_hwdata [31:0]	0	-	写数据总线: 32 位写数据总线。
biu_pad_hburst [2:0]	0	-	突发传输指示信号: 指示传输是一次突发传输的一部分: 000: SINGLE;

			001: INCR; 010: WRAP4。
biu_pad_hsize[2:0]	0	-	传输宽度指示信号: 指示传输的数据宽度: 000: byte; 001: halfword; 010: word。
biu_pad_htrans[1:0]	0	00	传输类型表示信号: 指示当前总线周期的传输类型: 00: IDLE; 01: BUSY; 10: NONSEQ; 11: SEQ。
biu_pad_hwrite	0	0	读写表示信号: 指示当前 CPU 是读取总线数据还是写总线: 1: 指示是写总线传输; 0: 指示是读总线传输。
biu_pad_hprot[3:0]	0	-	保护控制信号: 指示当前总线周期进行的数据传输的特性: ***0: 取指令; ***1: 数据访问; **0*: 用户访问; **1*: 超级用户访问; *0**: Not bufferable; *1**: bufferable; 0***: Not cacheable; 1***: cacheable。
biu_pad_hbusreq	0	0	总线请求信号: 指示 CPU 请求总线的使用权。
biu_pad_hlock	0	0	锁定总线信号: 当 lock 申明时, CPU 独占有总线控制权, 没有其他的 bus master 可以占有总线。
pad_biu_hrdata[31:0]	I	-	读数据总线: 32 位读数据总线。
pad_biu_hready	I	-	传输完成指示信号: 有效时指示当前传输已完成, CPU 将总线置于待命状态。
pad_biu_hgrant	I	-	总线占用指示信号: 有效时指示 CPU 当前已占用总线。
pad_biu_hresp[1:0]	I	-	传输应答信号: 传输应答: 00: OKAY; 01: ERROR; 10: RETRY; 11: SPLIT。
pad_biu_tmiss_b <sup>610, 620</sup>	I	-	总线传输失配指示信号: 有效时, 表示当前总线传输发生传输失配错误。 不用该信号时, 需要接 1。
<b>复位控制信号:</b>			
pad_reset_rst_b	I	-	处理器复位信号: 低电平时, 初始化 C-SKY CPU 内部及调试模块, C-SKY CPU 采用异步复位方式。
<b>外部中断控制和低功耗握手信号:</b>			
biu_pad_te_b	0	1	外部传输控制信号: 指示所要读取的地址应该经过一个外部的 MMU 转译。其对应于 PSR (处理器状态寄存器) 的 TE 位, 可以通过对 PSR 的修改控制传输。
pad_biu_int_b	I	-	中断请求信号: 低电平时表示进行普通中断申请。
pad_biu_fint_b	I	-	快速中断请求信号: 低电平时表示进行快速中断申请。
pad_biu_intraw_b	I	-	异步唤醒信号: 不请求进入中断, 用于将 CPU 从低功耗模式中唤醒。这个信号会使 biu_pad_wakeup_b 和

			biu_pad_ipend_b 有效。
pad_biu_finraw_b	I	-	快速异步唤醒信号: 不请求进入中断，用于将 CPU 从低功耗模式中唤醒。这个信号会使 biu_pad_wakeup_b 和 biu_pad_ipend_b 有效。
biu_pad_ipend_b	O	1	异步唤醒确认信号: 指示异步唤醒请求，已经被处理器确认。
pad_biu_vec_b[6:0]	I	-	中断矢量序号信号: 提供 core 进行中断处理的向量号，如果 pad_biu_avec_b 为低电平，这个向量被忽略。
Pad_biu_avec_b	I	-	中断向量有效控制信号: 如果为低电平，则普通中断和快速中断的向量入口取默认值；否则，普通中断和快速中断的向量入口取外部的输入 (pad_biu_vec_b[6:0])。
<b>功耗管理信号:</b>			
biu_pad_1pmd_b[1:0]	O	11	低功耗模式状态信号: 当处理器执行 doze, stop 或 wait 指令时，biu_pad_1pmd_b[1:0] 被相应的改变： 00: STOP; 01: WAIT; 10: DOZE; 11: normal。
biu_pad_wakeup_b	O	1	处理器唤醒指示信号: 低电平表示 CPU 被唤醒且进入正常工作模式，这个信号将会在 pad_biu_inraw_b, pad_biu_finraw_b, pad_biu_bkrq_b[1:0], pad_biu_dbgrq_b 或 pad_had_jdb_req_b 设置之后被设置。
<b>状态和时钟信号:</b>			
p11_core_cpuc1k	I	-	内核工作时钟信号: 提供 CPU 内核工作的时钟。
p11_core_sysc1k	I	-	总线同步时钟信号: Biu 与 CPU 外部系统同步时钟。
biu_pad_pstat[3:0]	O	-	处理器状态指示信号: 指示内部单元的状态，在时序上与 biu_pad_sysc1k 是同步的，因此这个信号只有在处理器与系统的时钟比例为 1 (pad_biu_c1kratio[2:0]=0) 时才有意义。 0000: 指令在 RF stage 有效，并且 retire buffer 不满； 0001: 指令在 RF stage 有效，并且 retire buffer 已满； 0010: 处理异常； 0011: RF stage 的指令无效； 0101: RF stage 的指令是一条 load/store 指令，但 LSU 没有就绪； 1000: 发射指令（除 1dm/ stm/ 1dq/ stq/ 1rw/ br/ bsr/ bt/ bf/ rte/ rfi/ jmp/ jsr/ jmpi / jsri）； 1001: 发射 1dm/ stm/ 1dq/ stq； 1011: 发射 1rw； 1100: 发射 br/ bsr/ bt/ bf； 1101: 发射 rte 或 rfi； 1110: 发射 jmp 或 jsr； 1111: 发射 jmpi 或 jsri； 其余：保留。
pad_biu_c1kratio[2:0]	I	-	CPU 时钟与总线时钟比例信号: CPU 时钟频率 / 总线时钟频率 = pad_biu_c1kratio +1: 000: 1 比 1； 001: 2 比 1； 010: 3 比 1； 011: 4 比 1； 100: 5 比 1；

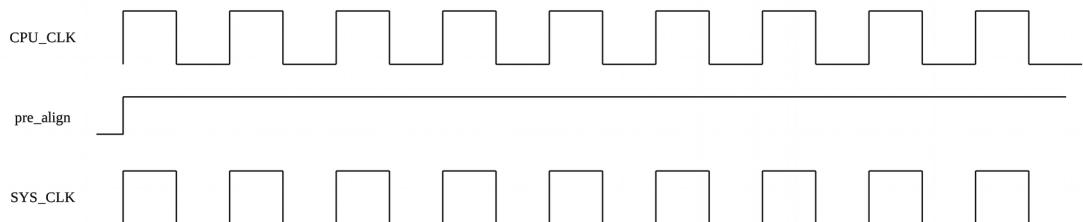
			101: 6 比 1; 110: 7 比 1; 111: 8 比 1。
biu_pad_idly4_b	0	1	idly4 信号指示信号: 指示一个 idly4 指令序列。当处理器执行了 idly4 这条指令之后，这个信号被清零，直到 IDLY4 指令序列被执行完才释放。这个信号与系统时钟同步，这就意味着当 CLK_RATIO 不等于 1(时钟比为 1:1)时，这个信号就不一一对应于 CPU 指令执行的情况，用户应根据具体情况深刻理解这信号与内部指令执行的对应关系。
pad_biu_bigend_b	I	-	指令解析模式指示信号: 低电平时表示取回来的数据或指令是 Big Endian 的字节顺序；这个信号为高电平时，表示取回来的数据或指令是 Little Endian 的字节顺序。该信号只有在 CPU 复位的时候才可以设置，一旦 CPU 退出复位状态，这个信号就保持静态输入，不能发生变化。
pad_biu_ext_align_en	I	-	该信号用于设置选择使用外部的采样对齐信号 (pad_biu_ext_pre_align)。 该信号为 1 时，采用外部信号 pad_biu_ext_pre_align 控制 CPU 对总线数据的采样。 该信号为 0 时，采用内部的 pre_align 信号来对总线数据进行采样
pad_biu_ext_pre_align	I	-	该信号提供给用户控制 CPU 总线采样的有效周期。CPU 将在该信号为 1 的 CPU 时钟对总线上的数据进行采样。具体参见 7.3 时钟部分的描述。 <b>注意：如果 pad_biu_ext_align_en 信号为 1 时，后端集成设计时，此信号的 setup 需要满足相应的要求，如 CPU 需要跑 500MHZ，此信号需要满足 500MHZ 对应的 setup</b>
<b>全局状态和控制信号：</b>			
biu_pad_gcb[11:0]	0	-	全局控制信号: 全局控制总线输出，用于控制外围设备和事件，它对应于 GCR 寄存器的低 12 位，因此可以通过 MTCR 修改 GCR 寄存器的值，达到控制外围设备的目的。当执行指令 MTCR 对全局控制寄存器进行更新时 biu_pad_gcb[11:0] 信号会改变。
pad_biu_gsb[11:0]	I	-	全局状态信号: 全局状态总线输入，用于检测外围设备和事件，它对应于 GSR 寄存器的低 12 位，但它只读。当执行指令 MFGR 将全局状态寄存器的值采样到一个通用寄存器中时，处理器对信号 pad_biu_gsb 进行采样。
<b>JTAG 支持信号：</b>			
pad_had_jtg_trst_b	I	-	JTAG 测试复位信号: JTAG 复位信号，下跳变可以初始化和 JTAG 接口相关的触发器，正常工作情况下该信号需置成高电平。
pad_had_jtg_tc1k	I	-	JTAG 测试时钟信号: JTAG 和 HAD 内部相关触发器的时钟信号，要求和 cpuc1k 的频率比在 1: 2 以下。
pad_had_jtg_tms	I	-	JTAG 测试模式选择信号: JTAG TAP 有限状态机状态跳转控制信号，该位可以控制命令的输入，数据的输入输出。
pad_had_jtg_tdi	I	-	JTAG 数据输入信号: JTAG 串行输入端口，包括控制命令，数据，输入顺序由低位到高位。
had_pad_jtg_tdo	0	-	JTAG 数据输出信号: JTAG 串行数据输出端口，输出顺序由低位到

			高位。
had_pad_jtg_tdo_en	0	-	TDO输出使能信号: 用于支持多个并行的 JTAG 控制器，当 tap 控制器处在 shift-dr 或 shift-ir 状态，并且 had_pad_jtg_tap_on 有效时，这个信号有效。
pad_had_jtg_tap_en	I	-	JTAG tap 控制器使能信号: 用于使能 JTAG tap 控制器，另外信号 pad_had_jdb_req_b 也可以达到同样的使能效果。
had_pad_jtg_tap_on	0	-	表明 tap 控制器状态指示信号: 用于指示 tap 控制器是否工作，高电平有效。
<b>调试支持信号:</b>			
pad_biu_dbgreq_b	I	-	外部调试请求信号: 该信号是外部调试请求信号，低电平有效，同步于 sysclk，使能处理器进入调试模式。 不用该信号时，需要接 1。
pad_biu_brkreq_b[1:0]	I	-	硬件断点请求: 该信号是外部硬件断点请求信号，低电平有效，可以使能处理器进入调试模式，同步于 sysclk。 不用该信号时，需要接 1。
biu_pad_dbg_b	0	1	表明处理器进入调试模式: 该位信号指示处理器是否处于调试模式，该信号低电平有效，同步于 sysclk。
pad_had_jdb_req_b	I	-	外部调试请求信号: 该信号是外部调试请求信号，异步于 tc1k，低电平有效。所以在 CPU 处于低功耗状态或者复位状态时，该信号有效可以快速请求 CPU 进入调试模式。 不用该信号时，需要接 1。
had_pad_jdb_ack_b	0	1	处理器响应进入调试模式: 处理器进入调试模式后，该响应信号保持两个 tc1k 的低电平。
had_pad_jdb_pm[1:0]	0	00	处理器工作模式指示信号: 这些输出信号表明处理器的工作模式，异步于 pad_had_jtg_tc1k。 00: normal 模式； 01: STOP/DOZE/WAIT 模式； 10: 调试模式； 11: 保留。
<b>测试支持信号:</b>			
pad_yy_test_mode	I	-	进入测试模式: 使处理器进入测试模式，此时的 cpu 时钟和系统时钟均为测试时钟 (pad_had_jtg_tc1k)，只有处理器进入测试模式，并且 pad_yy_scan_enable 有效时，才可以通过扫描链进行测试。 不用该信号时，需要接 0。
bz_core_si_12... bz_core_si_1	I	-	扫描链 scan in
bz_core_so_12... bz_core_so_1	0	-	扫描链 scan out
pad_yy_scan_enable	I	-	扫描使能: 扫描链的使能控制信号，高电平有效。 不用该信号时，需要接 0。
pad_yy_gate_clk_en_b	I	-	门控时钟使能信号: 处理器的门控时钟使能控制信号，只有当这个信号有效时，cpu 的内部模块的门控时钟才能有效。 不用该信号时，需要接 1。
<b>Memory Bist 信号:</b>			
*_smbist_done	0	0	bist 测试完成信号: 0: 表明片上 memory 自测试未完成； 1: 表明片上 memory 自测试完成。
*_smbist_fail	0	0	bist 测试失败信号:

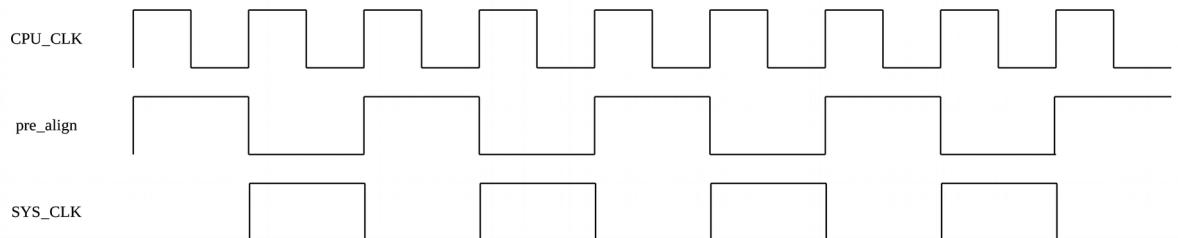
			0: 表明片上 memory 自测试成功, memory 工作正常; 1: 表明片上 memory 自测试失败。
bist_tst_en	I	-	bist 测试使能信号: 不用该信号时, 需要接 0。
mmu_da_smbist_done <sup>610M</sup>	0	0	mmu bist 测试完成信号: 0: 表明 mmu 的 memory 自测试未完成; 1: 表明 mmu 的 memory 自测试完成。
mmu_da_smbist_fail <sup>610M</sup>	0	0	mmu bist 测试失败信号: 0: 表明 mmu 的 memory 自测试成功, memory 工作正常; 1: 表明 mmu 的 memory 自测试失败。

### 7.3. 内部时钟与外部时钟的对齐

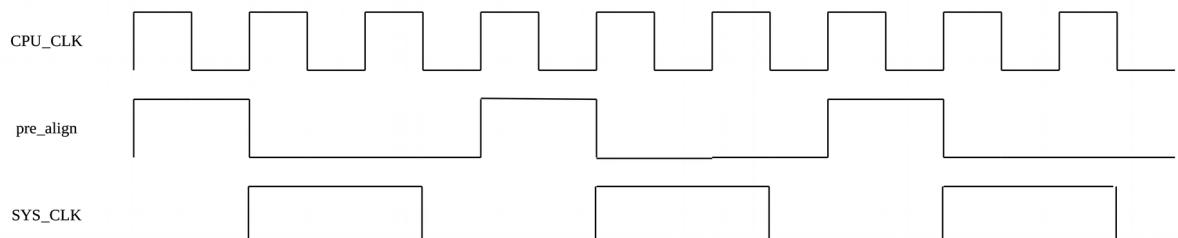
在 CK610 的设计中，CPU 内核时钟域与系统总线时钟域在总线接口部分逻辑中有信号交互。这两个时钟要求在物理设计中，需要保证时钟沿对齐。由于这两个信号只有整数倍关系，所以无需进行专用的逻辑进行同步，只需要一个逻辑信号控制 CPU 内部采样。在 CK610 的设计中，CPU 内部产生特定的控制信号（`pre_align`），保证 CPU 内部到总线上的所有信号都按照系统时钟的沿对齐。因此从 CPU 总线到系统的信号都具有 `sysclk` 的全周期时序延时。如果用户对这个 `pre_align` 有特殊的控制需求，CK610 提供了外部对齐的模式，即将 `pad_biu_ext_align_en` 设置为 1，CPU 在 `pad_biu_ext_pre_align` 的控制下对数据进行采样。这个 `pre_align` 信号在不同分频比例下的波形如下图所示。



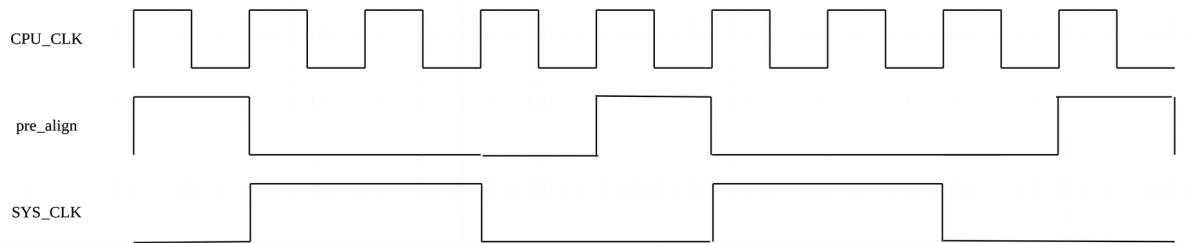
图表 接口信号-80CPU\_CLK 和 SYS\_CLK 1:1 波形图



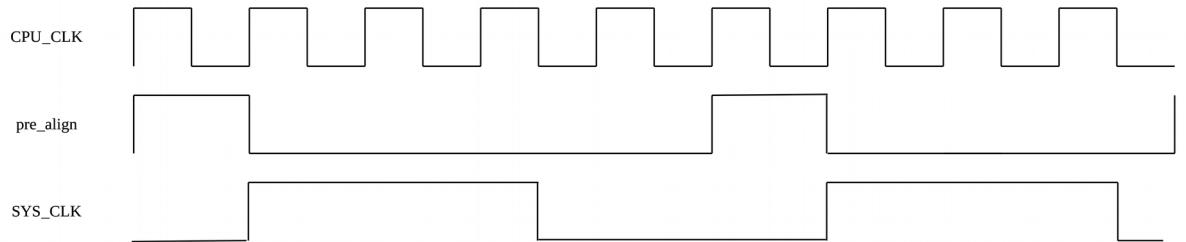
图表 接口信号-81 CPU\_CLK 和 SYS\_CLK 2:1 波形图



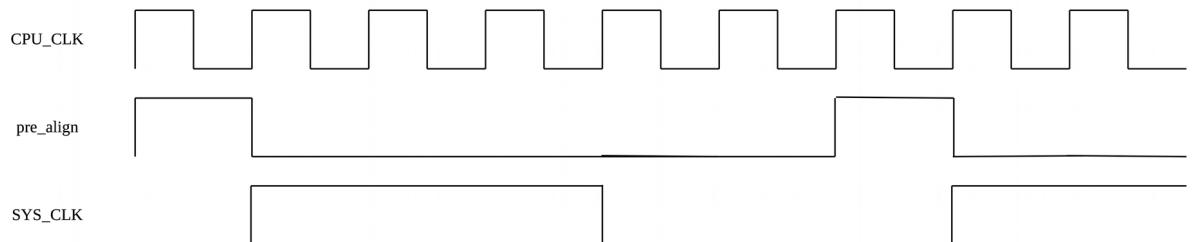
图表 接口信号-82 CPU\_CLK 和 SYS\_CLK 3:1 波形图



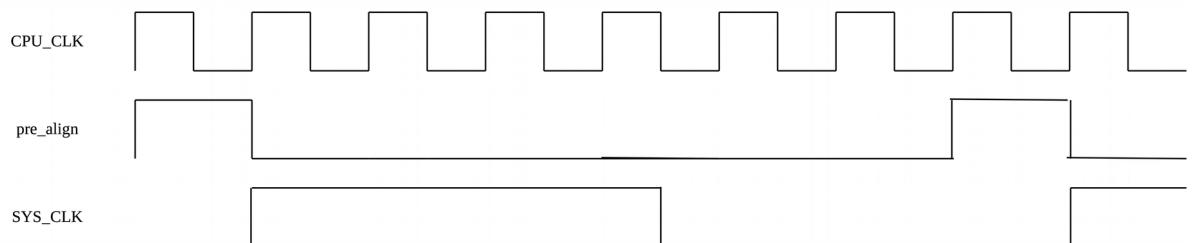
图表 接口信号-83 CPU\_CLK 和 SYS\_CLK 4:1 波形图



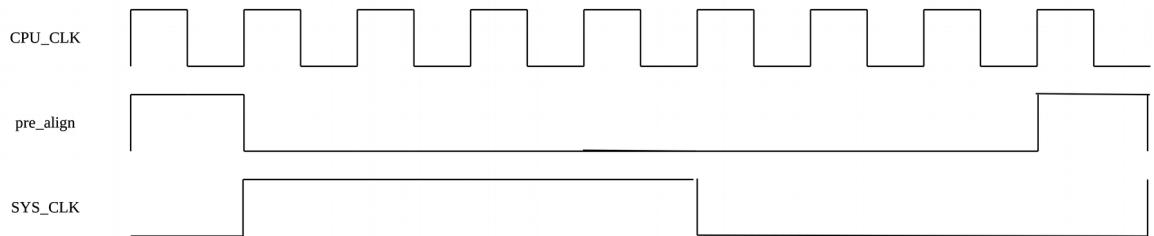
图表 接口信号-84 CPU\_CLK 和 SYS\_CLK 5:1 波形图



图表 接口信号-85 CPU\_CLK 和 SYS\_CLK 6:1 波形图



图表 接口信号-86 CPU\_CLK 和 SYS\_CLK 7:1 波形图



图表 接口信号-87 CPU\_CLK 和 SYS\_CLK 8:1 波形图

## 8. 接口总线协议

本章描述了 C-SKY CPU 的处理器接口总线协议。C-SKY CPU 的总线支持在处理器和其他外围设备之间同步传输数据。包括数据存取、地址控制和总线控制。

### 8.1. 外部接口信号列表

图表 接口总线协议-88 列出了 C-SKY CPU 处理器外部接口信号。

图表 接口总线协议-88 外部接口信号描述列表

信号名	简称	I/O	注释
<b>数据传输相关信号:</b>			
biu_pad_haddr[31:0]	HADDR	O	32位地址总线。
biu_pad_hwdata[31:0]	HWDATA	O	32位写数据。
biu_pad_hburst[2:0]	HBURST	O	指示传输是否 burst 传输及 burst 类型: 000:SINGLE Single transfer; 001:INCR Incrementing burst of unspecified length; 010:WRAP4 4-beat wrapping burst。
biu_pad_hsize[2:0]	HSIZE	O	指示传输的尺寸。
biu_pad_htrans[1:0]	HTRANS	O	指示传输类型, 应当是 00:IDLE, 01:BUSY, 10:NONSEQ, 11:SEQ 中的一种。
biu_pad_hwrite	HWRITE	O	1—指示是写传输; 0—指示是读传输。
biu_pad_hprot[3:0]	HPROT	O	保护控制信号。
biu_pad_hbusreq	HBUSREQ	O	总线请求信号。
biu_pad_hlock	HLOCK	O	表示总线已被CPU锁定。
pad_biu_hrdata[31:0]	HRDATA	I	32读数据。
pad_biu_hready	HREADY	I	指示当前传输完成。
pad_biu_hgrant	HGRANT	I	指示CPU当前已占有总线。
pad_biu_hresp[1:0]	HRESP	I	传输应答: 00: OKAY, 01: ERROR, 10: RETRY 和 11: SPLIT。
pad_biu_tmiss_b	TMISS	I	指示当前传输TLB未命中 miss。
<b>控制相关信号:</b>			
pad_biu_brkrq_b[1:0]	BRKRQ	I	一次访问的 breakpoint exception, 对应当前数据传输。
biu_pad_idly4_b	IDLY4	O	表示 IDLY4 状态。
biu_pad_pstat[3:0]	PSTAT	O	处理器状态输出, 具体见 Error: Reference source not found。
pad_biu_int_b	INT	I	中断输入。
pad_biu_fint_b	FINT	I	快速中断输入。
pad_biu_avec_b	AVEC	I	要求内部产生中断向量。
pad_biu_vec_b[6:0]	VEC	I	中断向量。
biu_pad_lpmd_b[1:0]	LPMD	O	表示低功耗模式: 00-stop; 01-wait; 10-doze; 11-normal。
biu_pad_gcb[31:0]	GCR	O	全局控制总线输出。
pad_biu_gsb[31:0]	GSR	I	全局状态总线输入。
pad_biu_dbgrq_b	DBGRQ	I	外部设备申请进入 debug 模式。
biu_pad_dbg_b	DBG	O	表示处理器已进入 debug 模式。
pad_biu_bigend_b	BIGEND	I	0 表示 big-endian; 1 表示 little-endian。
pad_biu_intraw_b	INTRAW	I	异步中断请求输入, 用于退出低功耗模式。
pad_biu_fintraw_b	FINTRAW	I	快速异步中断请求输入, 用于退出低功耗模式。
<b>时钟与复位信号:</b>			
pad_biu_sysclk	SYSCLK	I	从板上输入的系统时钟。
pad_reset_rst_b	RST	I	从板上输入的复位信号。
pad_biu_c1kratio[2:0]	CLKRATIO	I	表示处理器时钟频率和系统时钟频率的比例, 000: 1 比 1; 001: 2 比 1; 011: 4 比 1; 111: 8 比 1; 其它保留。

注意: I 表示输入, O 表示输出, 带有\_b 后缀的表示低电平有效。

## 8.2. 系统总线传输协议

C-SKY CPU 的外部总线传输协议和 AMBA2.0 的 AHB 兼容，所以我们可以参考 AMBA 2.0 规格说明（AMBA™ Specification Rev 2.0）。

### 8.2.1. 概览

总线接口单元负责 CPU Core 和 AHB 之间的地址控制和数据传输，其基本特点有：

- 与 AMBA2.0 总线协议兼容；
- 突发（burst）传输模式（4 word 回绕突发传输和未定长度增量连续的突发传输）；
- 关键字优先；
- Core-to-bus 时钟频率比可有不同配置；
- 当执行 idly4 指令时锁住总线；
- 所有的信号都 flopped in 或 flopped out 以获得较好的时序；
- 只有时钟上升沿时有操作；
- 没有三态执行；
- 支持 RETRY 和 SPLIT（可扩展）响应。

### 8.2.2. C-SKY CPU burst 传输

根据 C-SKY CPU 自身的特点，其外部系统总线传输协议和 AMBA 协议有一些不同点，在此说明如下：

- C-SKY CPU 只支持 4 拍回绕突发传输（4-beat wrapping burst）和未定长度增量突发传输（unspecified length incremental burst）；
- 4 拍回绕突发传输的宽度只支持 word；未定长度增量突发传输模式支持 byte, halfword, word。

**注意：**只有 CPU 执行 IDLY4 指令时 hlock 信号有效。如果主设备在 4 拍回绕突发传输或未定长度增量突发传输中收到 ERROR 响应，它会终止传输。

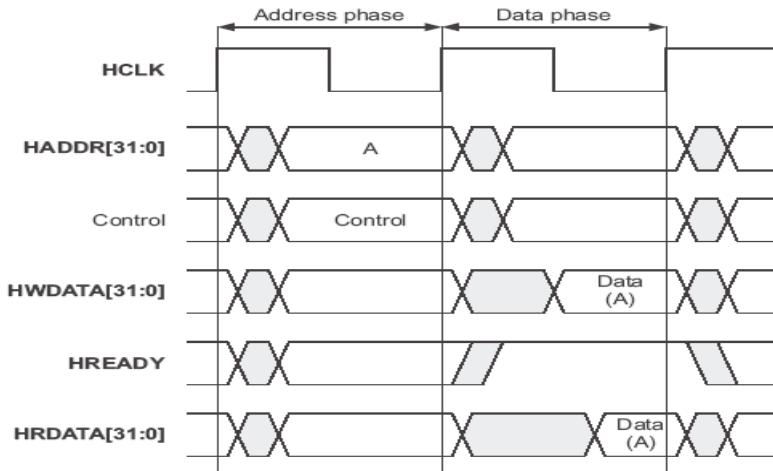
### 8.2.3. 一些典型的传输模式

CPU 发出 HBUSREQ 请求，等仲裁器给 CPU HGRANT 应答后，CPU 就占有了总线，可以向从设备发起传输。一个 AHB 传输包括两个不同的阶段：

- address phase：只持续一个时钟周期；
- data phase：通过使用 HREADY 信号可以持续多个时钟周期。

#### 8.2.3.1. 基本传输

图表 接口总线协议-89 简单传输表示了一个最简单的传输，没有等待状态。



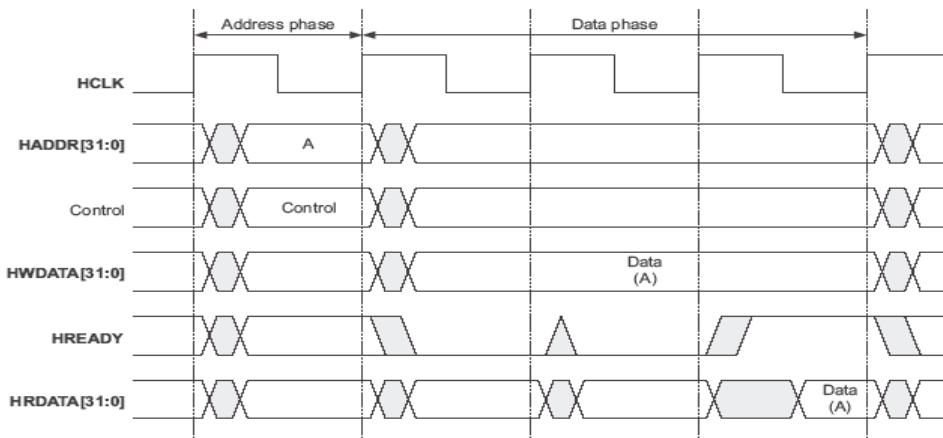
图表 接口总线协议-89 简单传输

在一个没有等待状态的简单传输中：

- 主设备（C-SKY CPU，下同）在 HCLK 的上升沿后向总线驱动地址和控制信号。
- 从设备在下一个 clock 的上升沿时获取地址和控制信息。
- 之后次设备开始发出合适的响应，总线主设备在 clock 的第三个上升沿获取这个响应。

实际上，任何传输的 address phase 在前一次传输的 data phase 就已产生了。这个地址和数据的重叠是这个总线的流水线特性的基础以获得高性能操作，同时仍可提供充分的时间给次设备发出传输的响应。

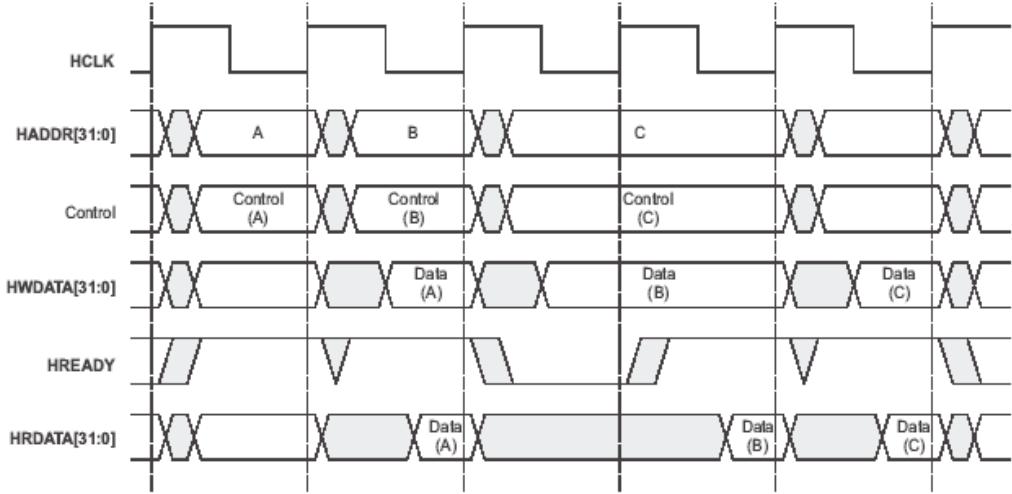
一个次设备可以向任何传输插入等待状态，如图表 接口总线协议-90 所示，延长了传输完成的时间。



图表 接口总线协议-90 有等待状态的传输

对于写操作主设备会在扩展的时钟周期里保持数据稳定，对于读传输次设备不需要提供有效数据直到传输快要完成。

当传输以这种方法扩展时它会产生扩展下一个传输的 address phase 的副作用。图表 接口总线协议-91 展示了对不相关的 A, B, C, 三个地址的传输。



图表 接口总线协议-91 多重传输

在图表 接口总线协议-91 中，向地址 A 和 C 的传输是零等待状态，向地址的传输是有一个等待状态，扩展向地址 B 的传输的 data phase 扩展了向地址 C 传输的 address phase。

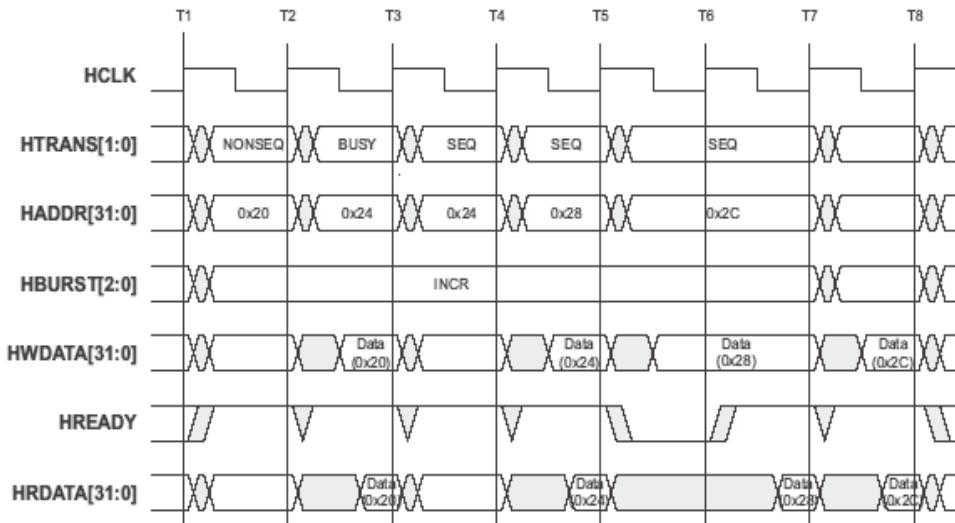
### 8.2.3.2. 传输类型

每次传输可以归类于四种不同传输类型中的一种，由 HTRANS[1:0]信号指示，具体见图表 接口总线协议-92。

HTRANS[1:0]	类型	描述
00	IDLE	指示此时没有数据传输被请求。当总线主设备已占有总线但还没有数据传输时，便是 IDLE 传输类型。次设备必须总是提供一个零等待状态 OKAY 响应给 IDLE 传输，并且这个传输应被次设备忽略。
01	BUSY	BUSY 传输类型允许总线主设备在突发传输中插入 IDLE 周期。这种传输类型指示总线主设备在继续一个突发传输中，但是下一个不能立即发生。当主设备使用 BUSY 传输类型地址和控制信号必须反映出突发传输模式中的下一个传输。这种传输应被次设备忽略。次设备必须总是提供一个零等待状态 OKAY 响应，和在 IDLE 传输中的情况一样。
10	NONSEQ	指示突发传输中的第一个传输或者一个单个传输。地址和控制信号和前次传输无关。总线上的单个传输被认为单个的突发传输，所以传输类型是 NONSEQUENTIAL。
11	SEQ	突发传输模式里剩下的传输是 SEQUENTIAL 并且地址和先前的传输相关。控制信息和先前的传输相同。address 等于先前的 address 加上传输 size (以字节单位)。在回绕突发传输模式中传输的地址在地址界限处回绕。

图表 接口总线协议-92 传输类型编码

图表 接口总线协议-93 表示了几种不同传输类型的使用。



图表 接口总线协议-93 传输类型实例

在图表 接口总线协议-93 中：

- 第一个传输是一个突发传输模式的开始，所以是 NONSEQUENTIAL 的。
- 主设备不能立即执行突发传输模式的第二次传输，所以主设备使用一个 BUSY 模式传输以延迟第二次传输的开始。在这个例子中主设备在开始突发传输的第二次传输前只需要一个时钟周期，没有等待状态就完成了。
- 主设备立即执行了突发传输模式的第三次传输，但此时从设备不能完成，使用 HREADY 有效以插入一个等待状态。
- 突发传输的最后一次传输以零等待状态完成。

### 8.2.3.3. 传输响应

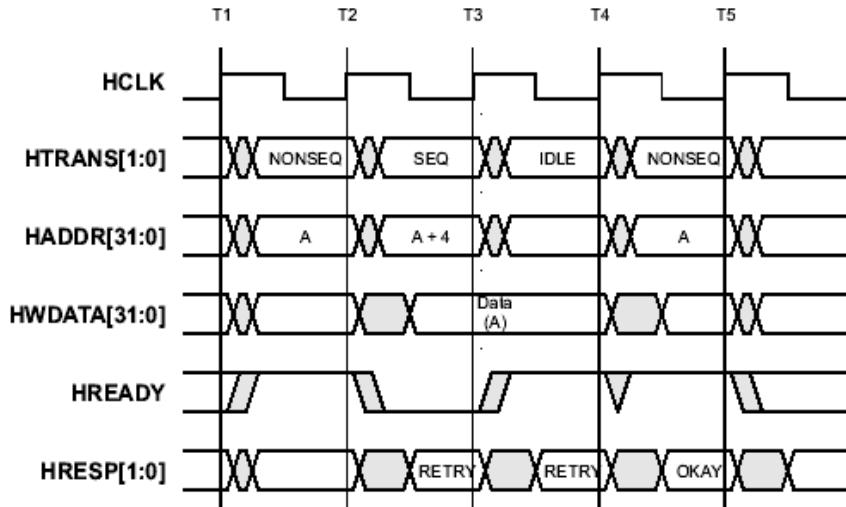
CPU 开始一次传输之后，次设备决定传输如何进行。每当次设备被访问他必须提供一个响应指示传输的状态。HREADY 信号用来扩展传输，它和给 CPU 传输的响应的 HRESP[1:0]一起工作。

HRESP[1:0]	响应类型	描述
00	OKAY	当 HREADY 变高 OKAY 表示传输成功完成。OKAY 循环也用在任何插入增加的循环，先于其他三个响应给出。
01	ERROR	这个响应表示有一个错误发生了。错误状态需要告知 CPU 使之知道传输不成功。ERROR 响应需要保持两个时钟周期。
10	RETRY	RETRY 响应表示传输尚未完成，所以 CPU 需要重试这次传输直到完成传输。RETRY 响应需要保持两个时钟周期。
11	SPLIT	传输尚未成功完成。CPU 当它下次被授予总线访问权时必须重试传输。当传输能完成的时候，次设备将会代表主设备请求对总线的访问。SPLIT 响应需要保持两个时钟周期（对此响应的操作 C-SKY CPU 可扩展）。

图表 接口总线协议-94 HRESP[1:0]响应类型描述

只有 OKAY 响应可只在单个周期给出。ERROR, SPLIT 和 RETRY 响应需要至少两个周期。为完成这些响应在倒数第二周期次设备驱动 HRESP[1:0]指示 ERROR, SPLIT 或 RETRY 同时驱动 HREADY 为低电平以扩展一额外的传输周期。最后驱动 HREADY 为高电平以结束传输同时响应置为 OKAY。

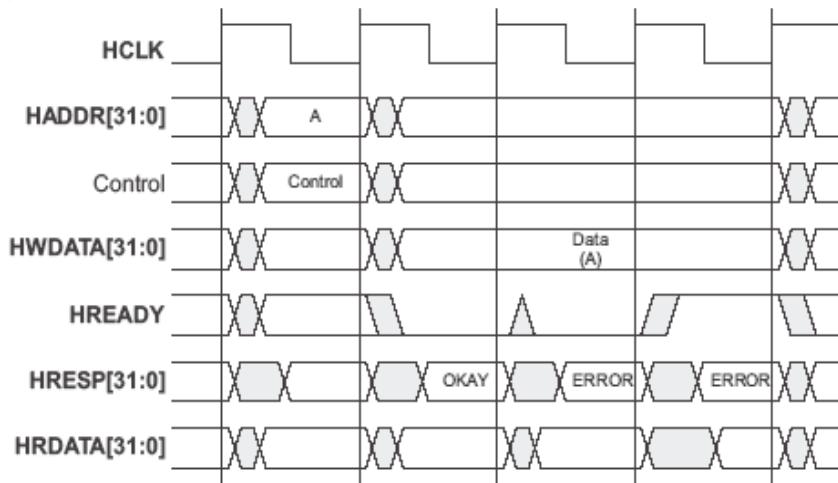
对于 SPLIT 和 RETRY 接下来的传输必须被取消。对于 ERROR，由于进行中的传输不被重复，是否进行接下来的传输是可选择的。



图表 接口总线协议-95 具有重传的传输

图表 接口总线协议-95 表示了具有重传的传输。

- CPU 开始传输于地址 A。
- 这次传输的响应接收到之前 CPU 将地址变为 A+4。
- 次设备在地址 A 上不能立即完成传输它发射了一个 RETRY 响应。这个响应告诉 CPU 在地址 A 的传输不能完成，所以在地址 A+4 的传输被取消，以一个 IDLE 传输代之。



图表 接口总线协议-96 错误响应

图表 接口总线协议-96 表示在一次传输中次设备需要一个周期决定发出那个响应（这时 HRESP 指示 OKAY）然后次设备以两个周期的 ERROR 响应结束了传输。CPU 收到 ERROR 响应后将跳到异常服务程序里。

#### 8.2.3.4. 突发传输操作

HBURST[2:0]指示了传输是否是 burst 传输及 burst 类型：

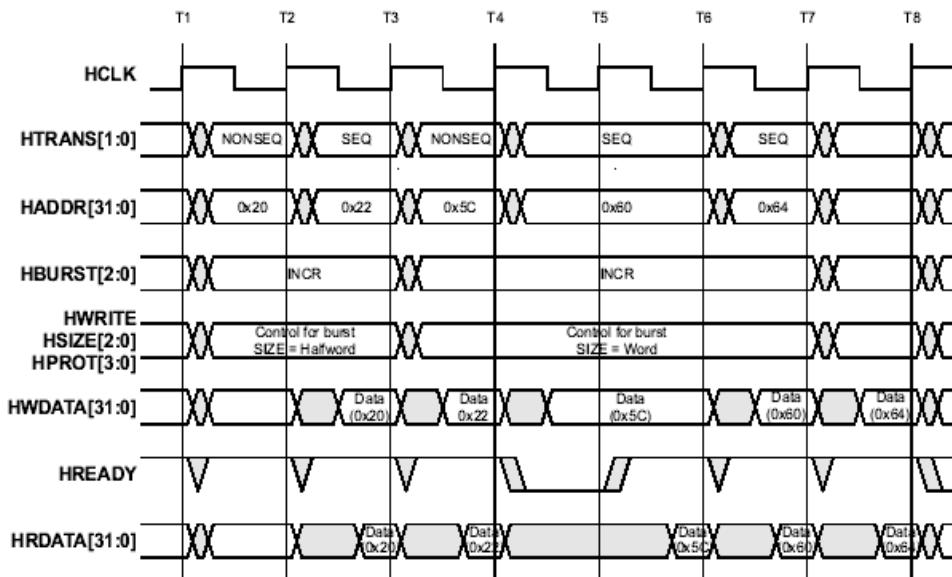
- 当 HBURST[2:0]为 000 时，代表单个传输（SINGLE）；
- 当 HBURST[2:0]为 001 时，代表未定长度增量突发传输（INCR）；
- 当 HBURST[2:0]为 010 时，代表四拍回绕突发传输（WRAP4）。

C-SKY CPU 只支持 INCR 和 WRAP4 两种突发传输模式。一个突发传输中的所有传输的地址必须和 HSIZEx 指示的每次传输大小指示的量对齐，比如以 word 为单位时地址的最后两位要是 0。Burst 模式访问地址范围不可一次跨越超过 1kB。当 C-SKY CPU 开了缓存运行时，每次取指令和访问数据都要取回 16Byte 的数据以 fill 一个 cache line，这时 CPU 就自动使用了 burst 传输模式。

某些特定情况下一个 burst 不能完成，把任何次设备设计成能如果 burst 提前终止利用 burst 信息进行正确操作是很重要的。次设备可以通过监视 HTRANS 信号和确认 burst 开始之后每次传输被标为 SEQUENTIAL 或 BUSY 知道 burst 是否已经提前结束。如果一个 NONSEQUENTIAL 或 IDLE 传输发生就指示了一个新的 burst 已经开始了，所以前面的那个一定已经结束了。

如果总线主设备由于失去了总线所有权不能完成一个 burst，当它下次能访问总线时必须以合适的方法重建 burst。例如，如果主设备只完成了 4 beat burst 中的 1 beat，然后它必须使用未定长度 burst 以执行剩下的 3 次传输。

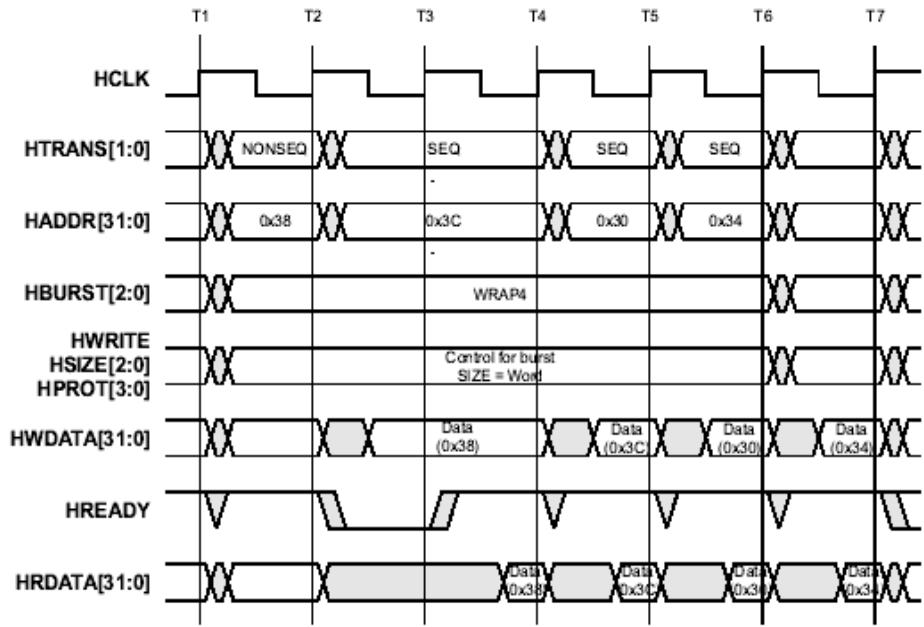
未定长度增量突发传输访问连续的位置，burst 中每次传输的地址是前一次传输地址的增量。一次增量突发传输可以是任意长的，上限是不可超过 1kB 地址范围。可以使用未定长度增量突发传输模式只执行一次传输以执行单个的传输。



图表 接口总线协议-97 增量突发传输

图表 接口总线协议-97 中，两次 halfword 的传输开始于地址 0x20，halfword 传输地址加 2。word 传输开始于地址 0x5C，地址增量是 4。

对于回绕突发传输，如果传输的开始地址和 burst 的总传输比特数不对齐，当传输地址达到此次 burst 地址界限时将回绕。对于 C-SKY CPU 来说，一个以 word (4 byte) 为单位的 4 beat wrapping burst 将在 16 byte 界限处回绕。就是说，如果如果传输的开始地址是 0x38，它包含的 4 个传输地址顺序就是 0x38, 0x3C, 0x30 和 0x34。如图表 接口总线协议-98 中所展示的那样。



图表 接口总线协议-98 四拍回绕突发传输

#### 8.2.4. 总线异常

图表 8-13 表示了总线上出现异常时 CPU 的行为：

图表 接口总线协议-99 总线异常控制循环

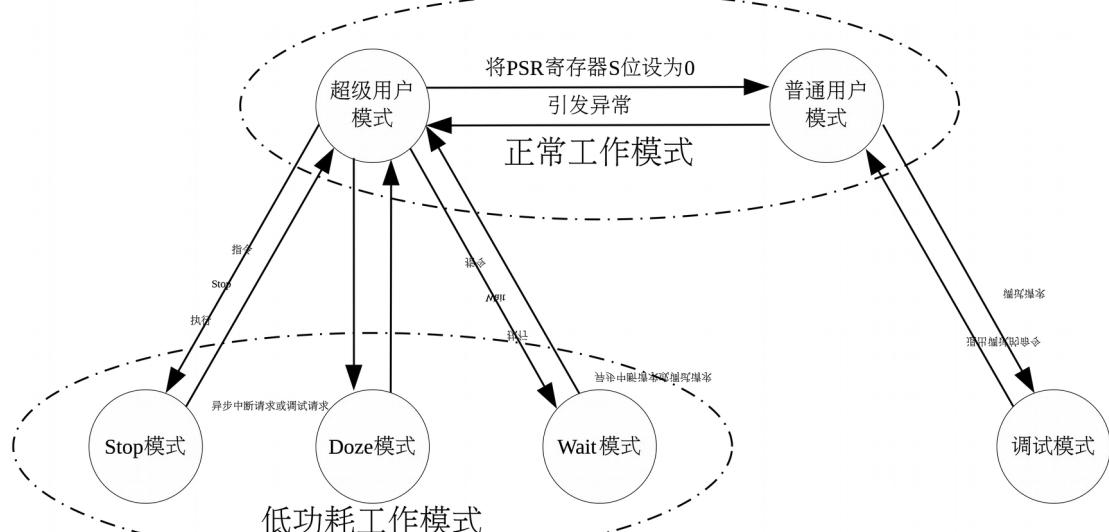
HREADY	HRESP	TMISS_b	结果
不关心	ERROR	High	访问错误-结束传输并处理访问错误。
不关心	ERROR	Low	TLB 丢失错误-结束传输并处理 TLB 丢失异常。
High	OKEY	不关心	正常循环结束并继续。
Low	OKEY	不关心	插入等待状态。
不关心	RETRY	不关心	等待进入 RETRY 操作。
不关心	SPLIT	不关心	等待进入分块传输。

## 9. 工作模式转换

CPU C-SKY CPU 总共有三类工作模式：正常运行模式、低功耗工作模式和调试模式，其中低功耗工作模式分为三种：STOP 模式、DOZE 模式、WAIT 模式，这三种低功耗模式的不同之处在于工作时停止的时钟不同。本章将详细解析 CPU 的工作模式以及模式之间的转换。

### 9.1. C-SKY CPU 工作模式及其转换

如图表 9-1 所示，CPU C-SKY CPU 的工作模式有三类，即正常工作模式、低功耗工作模式和调试模式，CPU 处于哪种工作模式可以通过查询 biu\_pad\_jdb\_pm[1:0]信号得到。



图表 工作模式转换-100 CPU 的各种工作状态示意图

#### 9.1.1. 正常工作模式

CPU 的正常工作模式可以分为两种，超级用户模式和普通用户模式，CPU 处于哪种正常工作模式通过查询 PSR 寄存器中的 S 位得到，当 S 位为 1 时，CPU 工作于超级用户模式；当 S 位为 0 时，CPU 工作于普通用户模式。当 CPU 工作在超级用户模式时，可以通过将 S 位设置为 0 进入普通用户模式；当 CPU 工作于普通用户模式时，通过引发异常进入超级用户模式。

#### 9.1.2. 低功耗模式

当 CPU 执行完低功耗指令(STOP、DOZE、WAIT)之后，CPU 将进入相应的低功耗模式。CPU 进入低功耗模式之后，CPU 的时钟被停止，只有异步中断请求 (pad\_biu\_intraw\_b 和 pad\_biu\_finraw\_b 信号) 或者调试请求才能正常地退出低功耗模式。当前的 CPU 工作于哪种低功耗模式可以通过查询信号线 biu\_pad\_lpmd\_b[1:0]来得到。

##### 9.1.2.1. DOZE 工作模式

当 CPU 执行完 DOZE 指令后，进入 DOZE 低功耗模式，此时 CPU 的时钟停止，哪些外设的时钟停掉与实现相关。

##### 9.1.2.2. STOP 工作模式

当 CPU 执行完 STOP 指令后，进入 STOP 低功耗模式，此时 CPU 和大多数外设的时钟

都被停止

### 9.1.2.3. WAIT 工作模式

当 CPU 执行完 WAIT 指令之后，进入 WAIT 低功耗模式，此时 CPU 停止工作，大多数的外设仍然在运行并可以产生中断。

## 9.1.3. 调试模式

### 9.1.3.1. 进入调试模式

当 CPU 接到调试请求之后，进入调试模式，其中的调试请求源可以有以下 7 种：

- 在低功耗模式下，可以通过使能 pad\_had\_jdb\_req\_b 信号进入调试模式。
- 在正常工作模式下，可以通过使能 pad\_biu\_dbgrq\_b 信号进入调试模式。
- 当 C-SKY CPU CSR 的 FDB 位有效时，pad\_biu\_brkrq\_b[1:0]信号有效进入调试模式。
- 当 C-SKY CPU CSR 的 FDB 位有效时，处理器在执行 bkpt 指令进入调试模式。
- 当 C-SKY CPU HCR 的 DR 位有效时，处理器在完成当前指令后进入调试模式。
- 当 C-SKY CPU HCR 的 TME 位有效时，处理器在跟踪计数器的值减到 0 后进入调试模式。
- 在 C-SKY CPU 存储器断点调试模式下，当 MBCA/MBCB 的值为 0 时，如果当前执行的指令符合断点要求，处理器进入调试模式。

### 9.1.3.2. 退出调试模式

如果 C-SKY CPU HACR 的 GO、EX 位被置为 1 时，并且当前的操作是对 CPUSCR 或者是对“没有选中任何寄存器”进行读写，则执行指令时 CPU 退出调试模式，进入正常工作模式。

**注意：**由于在调试模式下 PC、CSR、PSR 是可变的，因此在退出调试模式时，CPUSCR 中的值必须是刚进入调试模式之时保存过的值。

## 附录 A MGU 设置示例

```
*****
* Function: An example of set C-SKY CPU MGU.
*           Enable/disable C-SKY CPU data/instruction cache.
* Memory space: 0x28000000 ~ 0x29000000(16MBytes).
*
* Id |      Memory Space      | Write | Read | Cache
* 0 | 0x00000000 ~ 0xFFFFFFFF | Yes   | Yes  | NO
* 1 | 0x28000000 ~ 0x29000000 | Yes   | Yes  | Yes
* 2 | 0x28000000 ~ 0x28100000 | No    | Yes  | Yes
* 3 | 0x28F00000 ~ 0x29000000 | Yes   | Yes  | No
*
* Copyright (c) 2007 C-SKY Microsystems Co.,Ltd. All rights reserved.
*****/
```

```
/* Enable/disable every memory area. */
/* Set the access authorization. */
/* Set the cachable or not. */
lw      r10,0xef06
mtcr   r10,cr19

/* The first area (0x00000000 ~ 0xFFFFFFFF) */
movi   r10,0
mtcr   r10,cr21
lw      r10,0x3f          /* 4G space, Base address: 0x00000000 */
mtcr   r10,cr20

/* The second area (0x28000000 ~ 0x29000000) */
movi   r10,1
mtcr   r10,cr21
lw      r10,0x2800002f    /* 16M Space, Base address: 0x28000000 */
mtcr   r10,cr20

/* The third area (0x28000000 ~ 0x28100000) */
movi   r10,2
mtcr   r10,cr21
lw      r10,0x28000027    /* 1M Space, Base address: 0x28000000 */
mtcr   r10,cr20

/* The fourth area (0x28F00000 ~ 0x29000000) */
movi   r10,3
mtcr   r10,cr21
lw      r10,0x28f00027    /* 1M Space, Base address: 0x28F00000 */
mtcr   r10,cr20

/* Enable MGU */
mfcr   r7, cr18
bseti  r7, 0
bc1ri  r7, 1
mtcr   r7, cr18

/* Enable Branch prediction */
mfcr   r7 , cr18
bseti  r7 , 6
mtcr   r7 , cr18

/* Flush instruction cache */
movi   r7, 0x11
mtcr   r7, cr17

/* Enable instruction cache */
mfcr   r7, cr18
```

```
bseti    r7, 2
mtcr    r7, cr18

/* Flush Data cache */
movi    r7, 0x12
mtcr    r7, cr17

/* Enable data cache */
mfcr    r7, cr18
bseti    r7, 3
mtcr    r7, cr18
```

## 附录 B MMU 设置示例

```
*****  
* Function: An example of set C-SKY CPU's MMU TLB.  
*           Enable/disable C-SKY CPU data/instruction cache.  
* Memory space: Virtual address <-> physical address.  
*  
* virtual address: 0K-4K <-> physical address: 0K-4K  
* virtual address: 4K-8K <-> physical address: 4K-8K  
* virtual address: 8K-12K <-> physical address: 16K-20K  
* virtual address: 12K-16K <-> physical address: 8K-12K  
*  
* Copyright (c) 2007 C-SKY Microsystems Co.,Ltd. All rights reserved.  
*****/  
  
/*invalid all MMU TLB Entry*/  
movi    r1,0x0  
btsti   r1,26  
cpseti  15  
cpwcr   r1,cpcr8  
  
/* virtual address: 0K-4K <-> physical address: 0K-4K*/  
/* virtual address: 4K-8K <-> physical address: 4K-8K*/  
movi    r1,0x1e  
cpwcr   r1,cpcr2          //me10,PPN0:0x0  
1rw     r1,0x5e  
cpwcr   r1,cpcr3          //me11,PPN1:0x1  
movi    r1,0  
cpwcr   r1,cpcr4          //meh,VPN:0x0 & 0x1  
cpwcr   r1,cpcr6          //page size:4K  
bseti   r1,28  
cpwcr   r1,cpcr8          //write to jTLB  
  
/* virtual address: 8K-12K <-> physical address: 16K-20K*/  
/* virtual address: 12K-16K <-> physical address: 8K-12K*/  
movi    r1,0x11e  
cpwcr   r1,cpcr2          //me10,PPN0:0x4  
1rw     r1,0x9e  
cpwcr   r1,cpcr3          //me11,PPN1:0x2  
1rw     r1,0x2000  
cpwcr   r1,cpcr4          //meh,VPN:0x2 & 0x3  
movi    r1,0  
cpwcr   r1,cpcr6          //page size:4K  
bseti   r1,28  
cpwcr   r1,cpcr8          //write to jTLB  
  
/* enable mmu*/  
mfcr    r1,cr18  
bseti   r1,0x0  
mtcr    r1,cr18          //enable mmu
```

注意：使能MMU的这条mtcr语句所在的PC地址空间必须是映射过的，并已写入TLB，保证当MMU开启时，指令所在的PC地址空间可以完成异常，不出异常。

## 附录 B 指令术语表

附录 B 给出了每个 C-SKY CPU 指令的具体描述，下面根据缩写指令的英文字母顺序来对每条指令进行说明。

在指令格式中，寄存器 X、寄存器 Y、寄存器 Z 的编码表如下所示：

编码	寄存器	编码	寄存器
0000	R0	1000	R8
0001	R1	1001	R9
0010	R2	1010	R10
0011	R3	1011	R11
0100	R4	1100	R12
0101	R5	1101	R13
0110	R6	1110	R14
0111	R7	1111	R15

## ABS——绝对值指令

**操作:**  $RX \leftarrow |RX|$

**语法:** abs rx

**说明:** 取寄存器 X 的绝对值，并存储其计算结果。

注意，操作数0x80000000的结果80000000。

**影响标志位:** 无影响

**指令格式:**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	1	1	1	0				寄存器 X

## ADDC——无符号带进位 C 加法指令

- 操作：**  $RX \leftarrow RX + RY + C$ ,  $C \leftarrow$ 进位  
**语法：** addc rx, ry  
**说明：** 寄存器 X, 寄存器 Y 和进位标志位 C 的内容相加, 并把结果存在寄存器 X 中。  
**影响标志位：**  $C \leftarrow$ 进位  
**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0		寄存器 Y		寄存器 X				

## ADDI——无符号立即数加法指令

**操作：**  $RX \leftarrow RX + OIMM5$

**语法：** addi rx, oimm5

**说明：** 立即数与寄存器 X 内容相加。

**限制：** 立即数的范围为 1-32。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	0	0		OIMM5							寄存器 X

OIMM5 域——指定加到寄存器 RX 中的立即数。

注意：二进制数比起加到寄存器里的值需偏移 1。

00000——加 1

00001——加 2

.....

11111——加 32

## ADDU——无符号加法指令

**操作：**  $RX \leftarrow RX + RY$

**语法：** addu rx, ry

add rx, ry

**说明：** 寄存器 Y 的内容加到寄存器 X 中去，并把结果存在寄存器 X。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	0		寄存器 Y		寄存器 X				

## AND——按位与指令

**操作：** RX  $\leftarrow$  RX&RY

**语法：** and rx, ry

**说明：** 寄存器 X 与寄存器 Y 进行按位与，并把结果存储在寄存器 X 中。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	1	0		寄存器 Y		寄存器 X				

## ANDI——立即数按位与指令

**操作：**

$RX \leftarrow RX \& \text{无符号 IMM5}$

**语法：**

addi rx, imm5

**说明：**

立即数 IMM5 与寄存器 X 进行按位与操作，结果放在寄存器 X。

**限制：**

立即数的范围为 0-31。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	1		IMM5							寄存器 X

IMM5 域——指定无符号 5 位立即数的值。

00000——0

00001——1

.....

11111——31

## ANDN——按位与非指令

**操作：**  $RX \leftarrow RX \& (!RY)$

**语法：** andn rx, ry

**说明：** 寄存器 X 的值与寄存器 Y 的非值进行按位与操作，并把结果存在寄存器 X 中。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	1	1		寄存器 Y		寄存器 X				

## ASR——算术右移指令（动态）

- 操作：** RX  $\leftarrow$  RX 右移 RY[5:0]位。  
**语法：** asr rx, ry  
**说明：** 寄存器 RX 算术右移的位数等于 RY[5:0]的值，并把右移后的值存在 RX 中。如果 RY[5 : 0]的值大于 30，则根据 RX 的符号位，来决定 0 或 1 赋给 RX。  
**影响标志位：** 无影响  
**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	1	0		寄存器 Y		寄存器 X				

ASRC——算术右移一位到 C 进位位

**操作：**  $C \leftarrow RX[0]$ ,  $RX \leftarrow RX$  算术右移一位。

**语法：** asrc rx

**说明：**寄存器 RX 的最低位移入 C，其余向右移一位。

**影响标志位：** RX[0]在右移发生之前就被复制到 C 位。

### **指令格式：**

## ASRI——立即数算术右移（静态）

**操作：**

$RX \leftarrow RX$  右移 IMM5 位。

**语法：**

asri rx, imm5

**说明：**

$RX \leftarrow RX$  右移 IMM5 位 (1-31) ; 5 位立即数的值决定 RX 算术右移的位数。

**限制：**

立即数的范围为 1-31

**影响标志位：**

无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	0	1			IMM5						寄存器 X

IMM5 域——指定右移的位数值。

0001——1

0010——2

.....

1111——31

## BCLRI——立即数位清零

**操作：** 对 RX[IMM5]位清零。

**语法：** bclri rx, imm5

**说明：** 对 RX 的某个位清零，这个位由 IMM5 域的值决定。

**限制：** 立即数的范围为 0-31。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0			IMM5						寄存器 X

IMM5 域——指定为清零的位。

00000——位 0

00001——位 1

.....

11111——位 31

## BF——C 为 0 条件跳转指令

- 操作：** C 为 0 条件跳转：  
if(C==0), then  
    PC  $\leftarrow$  PC+2+(左移一位的有符号扩展的 11 位相对偏移量)；  
else  
    PC  $\leftarrow$  PC+2。  
**语法：** bf label  
**说明：** 如果 PSR 的 C 位为零时，PC  $\leftarrow$  PC+2+左移一位的有符号扩展的 11 位相对偏移量；  
否则，PC  $\leftarrow$  PC+2。即下一条指令的 PC 值。  
该偏移量是指 bf 指令下一条指令地址到目标地址的以半字为单位的长度。  
**影响标志位：** 无影响  
**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1											跳转位移域

跳转位移域——指定跳转位移范围。

## BGENI——立即数位产生指令（静态）

**操作：**

$RX \leftarrow (2)^{IMM5}$

**语法：**

bgeni rx, imm5

**说明：**

对由 IMM5 确定的寄存器 RX 位置位，并清除其他的寄存器位。

**限制：**

立即数的范围为 7-31；

立即数为 0 到 6 时，由 movi 指令来执行的。

**影响标志位：**

无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	0	1		IMM5							寄存器 X

IMM5 域——指定 RX 寄存器中被置位的寄存器位。

00111——位 7

01000——位 8

.....

11111——位 31

## BGENR——寄存器位产生指令

**操作：** if (RY[5] == 0) , then  
RX  $\leftarrow 2^{\text{RY}[4..0]}$ ;

else

RX  $\leftarrow 0_0$

**语法：** bgenr rx, ry

**说明：** 如果 RY[5]被清零，那么置位由寄存器 RY 的低五位（位 4：0）确定的 RX 寄存器位，并清除所有其他的位；否则，对 RX 清零。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	1	1		寄存器 Y		寄存器 X				

## BKPT——断点指令

**操作：** 引起一个异常断点指令或者进入调试模式

**语法：** bkpt

**说明：** 断点指令。

**影响标志位：**无影响

指令格式：

## BMASKI——立即数位屏蔽产生指令

**操作：** RX  $\leftarrow$  (2)<sup>IMM5-1</sup>

**语法：** bmaski rx, imm5

**说明：** 对寄存器 RX 的低 IMM5 位置 1，其余的高位清零。寄存器 RX 从 1 到 32 位都可以被置位。IMM5 域为 00000 时，汇编程序将认为是 32。

**限制：** 立即数的范围为 0、8-31；

立即数为 1-7 时由 movi 指令来执行的。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	0		IMM5							寄存器 X

IMM5 域——指定寄存器 RX 中要被置位的位数。

00000——0-31 位置位

01000——0-7 位置位

01001——0-8 位置位

.....

11111——0-30 位置位

## BR——无条件跳转指令

**操作：**  $PC \leftarrow PC + 2 + (\text{左移一位的有符号扩展的 } 11 \text{ 位相对偏移量})$

**语法：** br label

**说明：** 无条件跳转： $PC \leftarrow PC + 2 + \text{左移一位的有符号扩展的 } 11 \text{ 位相对偏移量。}$

该偏移量是指 br 指令下一条指令地址到目标地址的以半字为单位的长度。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	跳转偏移域										

跳转位移域——指定跳转位移范围。

## BREV——位倒序指令

- 操作：**把寄存器位的位置取倒序  
**语法：**brev rx  
**说明：**把寄存器 RX 位的位置取倒序。  
先前寄存器的值为”abcdefghijklmнопqrstuvwxyz012345”，取倒序后，它的值变为”543210zyxwvutsrqpонmlkjihgfedcba”。  
**影响标志位：**无影响  
**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	1	1	1				寄存器 X

## BSETI——立即数置位指令

- 操作：** 对寄存器 RX 的 imm5 位置 1  
**语法：** bseti rx, imm5  
**说明：** 对寄存器 RX 的某个位置 1，这个位由 IMM5 的值来确定。  
**限制：** 立即数的范围为 0-31。  
**影响标志位：** 无影响  
**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0			IMM5						寄存器 X

IMM5 域——指定寄存器需被置位的位。

00000——位 0

00001——位 1

.....

11111——位 31

## BSR——跳转到子程序指令

**操作：** 跳转到子程序：

$R15 \leftarrow PC+2$

$PC \leftarrow PC+2+(左移一位的有符号扩展的 11 位相对偏移量)$

**语法：** bsr label

**说明：** 返回地址存在寄存器 R15 中， $PC=PC+2+左移一位的有符号扩展的 11 位相对偏移量。$

该偏移量是指 bsr 指令下一条指令地址到目标地址的以半字为单位的长度。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	跳转偏移域										

跳转偏移域——指定跳转偏移范围。

## BT——C 为 1 条件跳转指令

**操作：** C 为 1 条件跳转：

```
if(C==1)
```

```
    PC ← PC+2+(左移一位的有符号扩展的 11 位相对偏移量)
```

```
else
```

```
    PC ← PC+2
```

**语法：** bt label

**说明：** 如果 PSR 的 C 位为 1 时， $PC \leftarrow PC+2+(左移一位的有符号扩展的 11 位相对偏移量)$ ；

否则， $PC \leftarrow PC+2$ 。即下一条指令的 PC 值。

该偏移量是指 bt 指令下一条指令地址到目标地址的以半字为单位的长度。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	跳转偏移域										

跳转位移域——指定跳转位移范围。

## BTSTI——立即数位测试指令

**操作：**  $C \leftarrow RX[IMM5]$

**语法：** btsti rx, imm5

**说明：** 对由 IMM5 决定的寄存器 RX 位进行测试，并使 C 位的值等于这一个位的值。

**限制：** 立即数的范围为 0-31。

**影响标志位：** 由 IMM5 决定的寄存器 RX 位决定。

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	1			IMM5						寄存器 X

IMM5 域——指定寄存器需被测试的位。

00000——位 0

00001——位 1

.....

11111——位 31

## CLRF——C 为 0 清零指令

- 操作：**当位 C 为 0 时，RX 被清零。  
**语法：**if(C==0), RX ← 0  
**说明：**当位 C 等于零时，寄存器 RX 被清零。  
**影响标志位：**无影响  
**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	1	1	0	1		寄存器 X		

## CLRT——C 为 1 清零指令

**操作：**当 C 为 1 时，RX 被清零

if(C==1), RX  $\leftarrow$  0

**语法：**clrt rx

**说明：**当位 C 等于 1 时，寄存器 RX 被清零。

**影响标志位：**无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	1	1	0	0	寄存器 X			

## CMPHS——大于等于比较指令

**操作：**寄存器X与寄存器Y作比较。

如果寄存器X的值大于等于寄存器Y的值，then

C ← 1；

else

C ← 0。

**语法：**cmphs rx, ry

**说明：**寄存器 X 的值减去寄存器 Y 的值，得到的值然后与 0 作比较，并对 C 位进行更新。cmphs 指令认为操作数是无符号数。如果 RX 大于等于 RY，则 C 位为 1；否则，C 位被清零。

**影响标志位：**由比较结果来对 C 位置位

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	0		寄存器 Y		寄存器 X				

## CMPLT——小于比较指令

**操作：**寄存器X与寄存器Y作比较。

如果寄存器X的值小于寄存器Y的值，then

C ← 1；

else

C ← 0。

**语法：**cmplt rx, ry

**说明：**寄存器 X 的值减去寄存器 Y 的值，得到的值然后与 0 作比较，并对 C 位进行更新。cmplt 指令认为操作数是有符号的补码形式的整数值。如果 RX 小于 RY，则 C 位为 1；否则，C 位被清零。

**影响标志位：**由比较结果来对 C 位置位

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1		寄存器 Y		寄存器 X				

## CMPLTI——立即数小于比较指令

**操作：** 寄存器X与立即数作比较。

如果条件为真, then

C  $\leftarrow$  1 ;

else

C  $\leftarrow$  0。

**语法：** cmplti rx, oimm5

**说明：** 寄存器 X 的值和立即数的值比较。cmplti 指令认为操作数是有符号的补码形式的整数值（立即数一般都认为是正数）。如果 RX 小于立即数的值，则 C 位为 1；否则，C 位被清零。

**限制：** 立即数的范围为 1-32。

**影响标志位：** 由比较结果来对 C 位置位

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	0	1		OIMM5							寄存器 X

OIMM5 域——指定跟寄存器 RX 比较的立即数的值。

注意二进制的编码值与比较的实际立即数的值会偏移 1。

00000——与 1 比较

00001——与 2 比较

.....

11111——与 32 比较

## CMPNE——不等比较指令

**操作：** 寄存器X与寄存器Y作比较。

如果条件为真, then

$C \leftarrow 1;$

    else

$C \leftarrow 0.$

**语法：** cmpne rx, ry

**说明：** 比较寄存器 X 的值与寄存器 Y 的值。

如果 RX 不等于 RY, 则 C 位为 1; 否则, C 位被清零。

**影响标志位：** 由比较结果来对 C 位置位

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1		寄存器 Y		寄存器 X				

## CMPNEI——立即数不等比较指令

**操作：**寄存器X与立即数作比较。

如果条件为真, then

C ← 1 ;

else

C ← 0。

**语法：**cmpnei rx, oimm5

**说明：**比较寄存器 X 与立即数的值。如果 RX 不等于立即数的值，则 C 位为

1；否则，C 位被清零。

**限制：**立即数的范围为 0-31。

**影响标志位：**由比较结果来对 C 位置位

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	0	1		OIMM5							寄存器 X

OIMM5 域——指定跟寄存器 RX 比较的立即数的值。

注意二进制的编码值与比较的实际立即数的值会偏移 1。

00000——与 0 比较

00001——与 1 比较

.....

11111——与 31 比较

## DECF——C 为 0 条件减 1 指令

**操作：** if (C==0), then  
          RX  $\leftarrow$  RX - 1 ;

        else  
          RX  $\leftarrow$  RX。

**语法：** decf rx

**说明：** 如果 C 位为零的话，寄存器 RX 减 1；否则寄存器的值不变。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	0	0	1		寄存器 X		

## DECGT——减 1 大于置位指令

**操作：**  $RX \leftarrow RX - 1$ , 更新C位

**语法：** decgt rx

**说明：** 寄存器 RX 减 1，如果其结果仍大于 0，则把 C 位置 1。

**影响标志位：** 如果寄存器减 1 结果大于 0，则 C 被置 1；否则 C 被清零。

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	1	0	1	0		寄存器 X		

## DECLT——减 1 小于置位指令

**操作：**  $RX \leftarrow RX - 1$ , 更新C位

**语法：** declt rx

**说明：** 寄存器 RX 减 1，如果其结果仍小于 0，则把 C 位置 1。

**影响标志位：** 如果寄存器减 1 结果小于 0，则 C 被置 1，否则 C 被清零。

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	寄存器 X

## DECNE——减 1 不等置位指令

**操作：**  $RX \leftarrow RX - 1$ , 更新C位

**语法：** decne rx

**说明：** 寄存器 RX 减 1，如果其结果仍不等于 0，则把 C 位置 1。

**影响标志位：** 如果寄存器 RX 减 1 结果不等于 0，则 C 被置 1，否则 C 被清零。

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	1	0	1	1		寄存器 X		

## DECT——C 为 1 条件减 1 指令

**操作 :** if (C==1), then  
RX  $\leftarrow$  RX - 1 ;

else  
RX  $\leftarrow$  RX。

**语法 :** dect rx

**说明 :** 如果 C 为 1 的话, 寄存器 RX 减 1; 否则寄存器的值不变。

**影响标志位 :** 无影响

**指令格式 :**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	寄存器 X

## DIVS——有符号寄存器除法指令

**操作：**

$$RX \leftarrow \frac{RX}{R1}$$

**语法：**

divs rx, r1

**说明：**

寄存器 RX 除以寄存器 R1，并把结果存在寄存器 RX 中，RX 和 R1 的值被认为是 32 位有符号的整数。对于 0x80000000 除以 0xFFFFFFFF 这种情况，结果没有定义。如果 R1 的值为 0，那么不能写回结果，并会产生一个被 0 处的异常中断。

**影响标志位：**无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	0	1	0	0	0	0	1				寄存器 X

## DIVU——无符号寄存器除法指令

**操作：**

$$RX \leftarrow \frac{RX}{R1}$$

**语法：**

divu rx, r1

**说明：**

寄存器 RX 除以寄存器 R1，并把结果存在寄存器 RX 中，RX 和 R1 的值被认为是 32 位无符号的整数。对于 0x80000000 除以 0xFFFFFFFF 这种情况，结果没有定义。如果 R1 的值为 0，那么不能写回结果，并会产生一个被 0 除的异常中断。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	0	0	0	0	0	1				寄存器 X

## DOZE——进入低功耗睡眠模式指令

**操作：**进入低功耗睡眠模式

**语法：** doze

**属性：**较高的优先级

**说明：**此指令使处理器进入低功耗睡眠模式，并等待一个中断来退出这个模式。此时，CPU 时钟停止，相应的外围设备也被停止。

**影响标志位：**无影响

#### **指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

## FF1——快速找 1 指令

**操作：**  $RX \leftarrow ff1(RX)$

**语法：** ff1 rx

**说明：** 查找寄存器 RX 中的第一个为 1 的位，并把查找结果返回到寄存器 RX。寄存器 RX 从最高位扫描到最低位。返回的值是偏移寄存器的最重要的位的位移。如果寄存器的位 31 被置 1，返回的值为 0。如果在寄存器 RX 没有为 1 的位，返回的值为 32。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	1	1	0				寄存器 X

## IDLY4——中断识别禁止指令

<b>操作：</b>	禁止中断识别四个指令
<b>语法：</b>	idly4
<b>说明：</b>	idly4 指令后四个指令禁止中断识别，这样就允许一个不可中断指令序列在多任务环境中被执行。
<b>影响标志位：</b>	标志位 C 在 idly4 指令执行后被清零。如果在 idly4 指令执行以后的四个指令中发生异常（包括跟踪或断点异常），位 C 被置 1，中断指令序列就能被观测到。
<b>限制：</b>	idly4 指令后面的指令只能是单时钟周期的算术、逻辑指令，ld, st, bf, 或 br 指令。为了使一些潜在中断的影响达到最小，如果是其他的指令，就不能保证不会被中断。如果任何一个其他指令被放在 idly4 序列里，那么不管什么样的异常，位 C 的结果值没有定义，而且指令序列有可能会被中断。如果在 idly4 后面的指令序列中有另一个 idly4 指令，那么它也将被忽略。但是如果是 rte, rfi, doze, wait, stop 等指令，那么他们就会引起 idly4 指令序列的中止。
<b>备注：</b>	<p>idly4 指令不允许在一个小于 8 条指令的循环中出现。 除非一个异常出现，不然的话，idly4 的计数器在使用 HAD 调试端口进行调试过程中不会变化。</p> <p>另外，如果 idly4 计数器停在不为零的状态时，中断就会被屏蔽。一旦处理器从调试模式中释放到正常的操作，那么计数就会继续。如果在 idly4 指令序列中发生一个断点异常或一个跟踪异常，那么位 C 会被置 1，这样序列将会出现操作失败。在异常处理过程中，使中断屏蔽无效，从而让计数器清零。这是跟踪或软件调试时，idly4 使用的一个限制。</p>

### **指令格式：**

## INCF——C 为 0 寄存器增 1 指令

**操作：** if C==0, then  
RX  $\leftarrow \text{RX} + 1$  ;

else  
RX  $\leftarrow \text{RX}$ 。

**语法：** incf rx

**说明：** 如果 C 为 0 的话，寄存器 RX 增 1；否则寄存器的值不变。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	0	1	1	1		寄存器 X	

## INCT——C 为 1 寄存器增 1 指令

**操作：** if C==1, then  
RX  $\leftarrow \text{RX} + 1$  ;

else  
RX  $\leftarrow \text{RX}$ 。

**语法：** inct rx

**说明：** 如果 C 为 1 的话，寄存器 RX 增 1；否则寄存器的值不变。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	0	1	0		寄存器 X		

## IXH——索引半字指令

**操作：** RX  $\leftarrow$  RX + [RY << 1]

**语法：** ixh rx,ry

**说明：** 寄存器 RY 的值左移一位后加上寄存器 RX 的值，然后把结果存入 RX。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	1		寄存器 Y		寄存器 X				

## IXW——索引字指令

**操作：** RX  $\leftarrow$  RX + [RY << 2]

**语法：** ixw rx,ry

**说明：** 寄存器 RY 的值左移两位后加上寄存器 RX 的值，然后把结果存入 RX。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	0	1		寄存器 Y		寄存器 X				

## JMP——无条件转移指令

**操作：** 无条件转移，

$PC \leftarrow \square(RX)$

**语法：** jmp rx

**说明：** 无条件的转移到寄存器 RX 决定的位置。RX 的最低位被忽略，并被清零。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	1	0	0				寄存器 X

## JMPI——无条件间接转移指令

- 操作：** 无条件间接转移，  
**语法：** PC  $\leftarrow \text{MEM}[(\text{PC}+2+(\text{unsigned disp\_8}<<2))\&0xffffffffc]$   
**说明：** 8位零拓展的位移域左移2位，并把它加到PC+2上，然后把生成的地址的后两位强制清零，并把由地址所取得的字存入PC。所以从本质上来说，目标地址被存储在和当前PC相关的的存储器位置，从目标地址取得的字存入PC，并在新的PC值继续执行指令。注意，转移目标地址相对于当前PC只有是向前偏移时，转移指令才是可行的。如果从目标地址所取得的值为奇数，那么就会发上一个未对准的异常中断。  
**影响标志位：** 无影响  
**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	0	0								DISP_8

DISP\_8 域——无符号8位位移范围。

## JSR——无条件子程序转移指令

**操作：**无条件子程序转移

$R15 \leftarrow PC+2, PC \leftarrow (RX)$

**语法：**jsr rx

**说明：**无条件转移到由寄存器 RX 决定的子程序中，并把返回的地址存在 R15 里。寄存器 RX 的最低位忽略并被清零。

**限制：**RX 不能是 R15。

**影响标志位：**无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1	1	0	1				寄存器 X

## JSRI——无条件子程序间接转移指令

**操作：**无条件子程序转移：

$R15 \leftarrow PC+2, PC \leftarrow MEM[(PC+2+(unsigned\ disp\_8<<2))\&0xffffffff]$

**语法：**jsri label

**说明：**PC+2 的值被存在寄存器 R15 中，8 位零拓展的位移域左移 2 位，并把它加到 PC+2 上，然后把生成的地址的低两位强制清零，并把由地址所取得的值存入 PC。所以从本质上来说，目标地址被存储在和当前 PC 相关的的存储器位置，从目标地址取得的字存入 PC，并在新的 PC 值继续执行指令。注意，转移目标地址相对于当前 PC 只有是向前偏移时，转移指令才是可行的。如果从目标地址所取得的值为奇数，那么就会发上一个未对准的异常中断。

**影响标志位：**无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1								DISP_8

DISP\_8 域——无符号 8 位位移范围。

## LD.[BHW]——从存储器装入寄存器

- 操作：** 源/目标寄存器  $\leftarrow$  存储器：  
 $RZ \leftarrow \text{MEM}[RX + \text{unsigned IMM4} \ll \{0,1,2\}]$
- 语法：** ld.[bhw] rz, (rx, disp)
- 说明：** 装入操作由三个选项：w(字)，h(半字)和 b (字节)。通过 4 位立即数装入内容的大小，并进行零拓展来获得偏移量。然后把偏移量与寄存器 RX 的值相加等于新的地址，从新的地址按照指定的大小读取数据，并把结果存入寄存器 RZ 中。对于字节和半字来说，索取的数据在被存入 RZ 之前，需要进行零拓展。
- 影响标志位：** 无影响
- 指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	SIZE	0	寄存器 Z			IMM4			寄存器 X						

SIZE——说明装入数据的大小。

00——字

01——字节

10——半字

IMM4——4 位立即数。

## LDM——从存储器装入多个寄存器

**操作：**

目标寄存器  $\leftarrow$  存储器

**语法：**

ldm rf-r15, (r0)

**说明：**

ldm 指令用于从存储器向一些邻近的寄存器装入数据。

**限制：**

寄存器 R0 将作为基址指针，寄存器 Rf-R15 依次从存储器装入数据。

**影响标志位：**

Rf 不能是 R0 或 R15。

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	1	1	0				寄存器 F

寄存器 RF——指定要被传输的第一个寄存器。

## LDQ——从存储器装入 4 个寄存器

**操作：**

目标寄存器  $\leftarrow$  存储器

**语法：**

ldq r4-r7, (rx)

**说明：**

ldq 指令用于从存储器向 4 个寄存器 (R4—R7) 装入数据。寄存器 RX 指定第一个被传输数据的位置。

**限制：**

RX 不能是 R4, R5, R6, R7。

**影响标志位：**

无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	1	0	0		寄存器 X		

寄存器 RX——指定要被传输的基址。

## LRW——从存储器读入

**操作：**

$RZ \leftarrow \text{MEM}[(\text{PC}+2+(\text{unsigned disp\_8}<<2))\&0xffffffffc]$

**语法：**

lrw rz, lable

lrw rz, imm32

**说明：**

DISP\_8 左移 2 位，并把它加到 PC+2 上，然后把生成的地址的低两位强制清零，并把由地址所取得的字存入寄存器 RZ。

**限制：**

RZ 不可以是 R0 或 R15。

**影响标志位：**无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1				寄存器 Z								DISP_8

DISP\_8 域——是汇编程序产生的，用于指向 IMM32 存放地址的 8 位偏移量。

## LSL——逻辑左移指令（动态）

- 操作：**  $RX \leftarrow RX$  逻辑左移  $RY[5:0]$  位  
**语法：** lsl rx, ry  
**说明：** 寄存器 RX 逻辑左移，左移位数由  $RY[5:0]$  的值决定；如果  $RY[5:0]$  的值大于 31，那么寄存器 RX 清零。  
**影响标志位：** 无影响  
**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	1	1		寄存器 Y		寄存器 X				

### **LSLC——逻辑左移 1 位指令**

**操作：**  $C \leftarrow RX[31]$ ,  $RX \leftarrow RX$ 逻辑左移1位

**语法：** lslc rx

**说明：** RX  $\leftarrow$  RX 逻辑左移 1 位，并把 RX[31] 移入位 C。

**影响标志位**： 在左移之前把 RX[31] 复制到位 C。

### **指令格式：**

## LSLI——立即数逻辑左移指令（静态）

- 操作：** RX  $\leftarrow$  RX 逻辑左移 IMM5 位  
**语法：** lsli rx, imm5  
**说明：** RX  $\leftarrow$  RX 逻辑左移 IMM5 位(1..31)位。  
**限制：** IMM5 必须为 1-31。  
**影响标志位：** 无影响  
**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	1	0		IMM5							寄存器 X

IMM5 域——指定左移的位数。

0001——1

0010——2

.....

1111——31

## LSR——逻辑右移指令（动态）

- 操作：**  $RX \leftarrow RX$  逻辑右移  $RY[5:0]$  位  
**语法：** lsr rx, ry  
**说明：** 寄存器 RX 逻辑右移，右移位数由  $RY[5:0]$  的值决定。如果  $RY[5:0]$  的值大于 31，那么寄存器 RX 清零。  
**影响标志位：** 无影响  
**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	1		寄存器 Y		寄存器 X				

**LSRC——逻辑右移 1 位指令**

**操作：**  $C \leftarrow RX[0]$ ,  $RX \leftarrow RX$ 逻辑右移1位

**语法：** lsric rx

**说明：**  $RX \leftarrow RX$  逻辑右移 1 位，并把  $RX[0]$  移入位  $C$ 。

**影响标志位**： 在右移之前把 RX[0] 复制到位 C。

## **指令格式：**

## LSRI——立即数逻辑右移指令（静态）

- 操作：** RX  $\leftarrow$  RX 逻辑右移 IMM5 位  
**语法：** lsri rx, imm5  
**说明：** RX  $\leftarrow$  RX 逻辑右移 IMM5 位(1..31)位。  
**限制：** IMM5 必须为 1-31。  
**影响标志位：** 无影响  
**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	1	1			IMM5						寄存器 X

IMM5 域——指定左移的位数。

0001——1

0010——2

.....

1111——31

## MAC<sup>620</sup>——乘法累加指令

**操作：**  $(HI, LO) \leftarrow (HI, LO) + RX \times RY$

**语法：** mac rx, ry

**说明：** 寄存器 RX 的值与寄存器 RY 的值相乘，然后再加上 64 位 HI/LO 寄存器中的数据，并把结果的低 32 位存入 LO 寄存器中，高 32 位存入 HI 寄存器中。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0		寄存器 Y		寄存器 X				

## MFCR——控制寄存器数据读传送指令

**操作：**  $RX \leftarrow CRy$

**语法：** mfcr rx, cry

**属性：** 较高优先权。

**说明：** 控制寄存器 Y 中的数据传到寄存器 RX 中。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0		控制寄存器 Y				寄存器 X			

控制寄存器 Y 域——指定源控制寄存器 CRy。

00000 —— 控制寄存器 CR0

00001 —— 控制寄存器 CR1

.....

11110 —— 控制寄存器 CR30

## MFHI<sup>620</sup>——高位寄存器数据读传送指令

**操作：** RX  $\leftarrow$  HI

**语法：** mfhi rx

**说明：** 高位寄存器 HI 的数据传送到寄存器 RX 中。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	0	0	0	1	1	0		寄存器 X		

## MFHIS<sup>610E</sup>——Hi 寄存器数据饱和值传送指令

**操作：** Rx  $\leftarrow$  saturation (Hi)

**语法：** MFHIS rx

**说明：** CK610E 为 32x16 和 16x16 乘累加实现了 8 位保护位。当多条 32x16 和 16x16 乘加或乘减指令连续执行时，结果可能溢出 32 位宽。使用该指令可以得到 Hi 寄存器中结果的饱和值。如果没有发生溢出，则执行结果和 MFHI 指令一致。该指令不影响 Hi 寄存器中的值。

**影响标志位：** 无影响

**溢出标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	0	0	0	0	1	1		寄存器 X		

## MFLO<sup>620</sup>——低位寄存器数据读传送指令

**操作：** RX  $\leftarrow$  LO

**语法：** mflo rx

**说明：** 低位寄存器 LO 的数据传送到寄存器 RX 中。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	0	0	0	1	1	1	1	1	1	寄存器 X

## MFLOS<sup>610E</sup>——Lo 寄存器数据饱和值传送指令

**操作：** Rx  $\leftarrow$  saturation (Lo)

**语法：** MFLOS rx

**说明：** CK610E 为 32x16 和 16x16 乘累加实现了 8 位保护位。当多条 32x16 和 16x16 乘加或乘减指令连续执行时，结果可能溢出 32 位宽。使用该指令可以得到 Lo 寄存器中结果的饱和值。如果没有发生溢出，则执行结果和 MFLO 指令一致。该指令不影响 Lo 寄存器中的值。

**影响标志位：** 无影响

**溢出标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	0	0	0	0	1	1		寄存器 X		

## MOV——逻辑数据传送指令

**操作：** RX  $\leftarrow$  RY

**语法：** mov rx, ry

**说明：** 把寄存器 Y 中的值复制到目标寄存器 RX 中。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	1	0		寄存器 Y		寄存器 X				

## MOVF——C 为 0 逻辑数据传送指令

**操作：** if (C==0),  
 RX  $\leftarrow$  RY。

**语法：** movf rx, ry

**说明：** 当位 C 为零时，把寄存器 Y 中的值复制到目标寄存器 RX 中。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0		寄存器 Y		寄存器 X				

## MOVI——立即数逻辑数据传送指令

**操作：** RX  $\leftarrow$  无符号7位立即数

**语法：** movi rx, imm7

**说明：** 把零扩展的7位立即数传送到目标寄存器 RX 中。

**限制：** 立即数的范围为 0-127。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0											寄存器 X

IMM7 域——指定传送到寄存器 RX 的立即数。

0000000——0

0000001——1

.....

1111111——127

## MOVT——C 为 1 逻辑数据传送指令

**操作：** if (C==1),  
 RX  $\leftarrow$  RY。

**语法：** movt rx, ry

**说明：** 当位 C 置 1 时，寄存器 RY 中的数据传送到 RX 中。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0		寄存器 Y		寄存器 X				

## MTCR——控制寄存器数据写传送指令

**操作：** CRy  $\leftarrow$  RX

**语法：** mtcr rx, cry

**属性：** 较高优先权。

**说明：** 寄存器 RX 的数据传送到控制寄存器 CRy 中。

**影响标志位：** 如果 CR0 (PSR) 不被指定的话，那么无影响。

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	0		控制寄存器 Y				寄存器 X			

控制寄存器 Y 域——指定源控制寄存器 CRy。

00000——控制寄存器 CR0

00001——控制寄存器 CR1

.....

11110——控制寄存器 CR30

## MTHI<sup>620</sup>——高位寄存器数据写传送指令

**操作：** HI ← RX

**语法：** mthi rx

**说明：** 寄存器 RX 的数据传送到高位寄存器 HI 中。

**影响标志位：** 无影响。

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	0	0	0	1	0	1	1	1	1	寄存器 X

## MTLO<sup>620</sup>——低位寄存器数据写传送指令

**操作：** LO  $\leftarrow$  RX

**语法：** mtlo rx

**说明：** 寄存器 RX 的数据传送到低位寄存器 LO 中。

**影响标志位：** 无影响。

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	0	0	0	1	0	0			寄存器 X	

## MULS<sup>610E</sup>——有符号 32 位乘法指令

- 操作：**  $(s)\{Hi[31:0] | Lo[31:0]\} \leftarrow (s)Rx[31:0] \times (s)Ry[31:0]$
- 语法：** muls rx, ry
- 说明：** 寄存器 X 的值和寄存器 Y 的值有符号相乘。积的低 32 位存入 Lo 寄存器，高 32 位存入 Hi 寄存器。
- 影响标志位：** 无影响。
- 溢出标志位：** 溢出位清零
- 指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	0	0		寄存器 Y		寄存器 X				

## MULSA<sup>610E</sup>——有符号 32 位乘累加指令

**操作：**  $(s)\{Hi[31:0] | Lo[31:0]\} \leftarrow (s)\{Hi[31:0] | Lo[31:0]\} + (s)Rx[31:0] \times (s)Ry[31:0]$

**语法：** mulsa rx, ry

**说明：** 寄存器 X 的值和寄存器 Y 的值有符号相乘，相乘的积和 {Hi[31:0]|Lo[31:0]} 相加。最终结果的低 32 位存入 Lo 寄存器，高 32 位存入 Hi 寄存器。该指令不支持保护位。

**影响标志位：** 无影响。

**溢出标志位：** 有影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	0	1		寄存器 Y		寄存器 X				

## MULSH——16位有符号乘法指令

- 操作：**  $RX \leftarrow RX[15\ldots0] \times RY[15\ldots0]$
- 语法：** mulsh rx, ry  
muls.h rx, ry
- 说明：** 寄存器 RX 的低 16 位乘以寄存器 RY 的低 16 位，并把有符号的结果存入寄存器 RX 中。16×16->32(有符号)。源操作数被认为是 2 个补码形式有符号的数，位 15 是符号位。
- 影响标志位：** 无影响
- 指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	0	0	0		寄存器 Y		寄存器 X				

## MULSHA<sup>610E</sup>——有符号 16 位乘累加指令

- 操作 :**  $(s)\{Lo[31:0]\} \leftarrow (s)\{Lo[31:0]\} + (s)Rx[15:0] \times (s)Ry[15:0]$
- 语法 :** mulsha rx , ry
- 说明 :** 寄存器 X 的低 16 位和寄存器 Y 的低 16 位有符号相乘，相乘的积和 Lo 寄存器的值相加。最终结果存入 Lo 寄存器。该指令支持 8 位的保护位。
- 影响标志位 :** 无影响
- 溢出标志位:** 有影响
- 指令格式 :**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	1		寄存器 Y		寄存器 X				

## MULSHS<sup>610E</sup>——有符号 16 位乘累减指令

- 操作：**  $(s)\{Lo[31:0]\} \leftarrow (s)\{Lo[31:0]\} - (s)Rx[15:0] \times (s)Ry[15:0]$
- 语法：** mulshs rx , ry
- 说明：** 寄存器 X 的低 16 位和寄存器 Y 的低 16 位有符号相乘，然后 Lo 寄存器的值减去相乘的积。最终结果存入 Lo 寄存器。该指令支持保护位。
- 影响标志位：** 无影响
- 溢出标志位：** 有影响
- 指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	1	0		寄存器 Y		寄存器 X				

## MULSS<sup>610E</sup>——有符号 32 位乘累减指令

**操作：**  $(s)\{Hi[31:0] | Lo[31:0]\} \leftarrow (s)\{Hi[31:0] | Lo[31:0]\} - (s)Rx[31:0] \times$

$(s)Ry[31:0]$

**语法：** mulss rx, ry

**说明：** 寄存器 X 的值和寄存器 Y 的值有符号相乘，然后  $\{Hi[31:0]|Lo[31:0]\}$  减去相乘的积。最终结果的低 32 位存入 Lo 寄存器，高 32 位存入 Hi 寄存器。该指令不支持保护位。

**影响标志位：** 无影响

**溢出标志位：** 有影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	1	0		寄存器 Y		寄存器 X				

## MULSW<sup>610E</sup>——有符号 16x32 乘法指令

- 操作：**  $(s)Rx[31:0] \leftarrow ((s)Rx[15:0] \times (s)Ry[31:0])[47:16]$
- 语法：** mulsw rx, ry
- 说明：** 寄存器 X 的低 16 位和寄存器 Y 的值有符号相乘，积的高 32 位存入寄存器 X。
- 影响标志位：** 无影响
- 溢出标志位：** 无影响
- 指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	0	0		寄存器 Y		寄存器 X				

## MULSWA<sup>610E</sup>——有符号 16x32 乘累加指令

- 操作 :**  $(s)\{Lo[31:0]\} \leftarrow (s)\{Lo[31:0]\} + (s)((s)Rx[15:0] \times (s)Ry[31:0])[47:16]$
- 语法 :** mulswa rx, ry
- 说明 :** 寄存器 X 的低 16 位和寄存器 Y 的值有符号相乘，积的高 32 位和 Lo 寄存器的值相加。最终结果存入 Lo 寄存器。该指令支持 8 位的保护位。
- 影响标志位 :** 无影响
- 溢出标志位:** 有影响
- 指令格式 :**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	0	1		寄存器 Y		寄存器 X				

## MULSWS<sup>610E</sup>——有符号 16x32 乘累减指令

- 操作 :**  $(s)\{Lo[31:0]\} \leftarrow (s)\{Lo[31:0]\} - (s)((s)Rx[15:0] \times (s)Ry[31:0])[47:16]$
- 语法 :** mulsws rx , ry
- 说明 :** 寄存器 X 的低 16 位和寄存器 Y 的值有符号相乘，然后 Lo 寄存器的值减去积的高 32 位。最终结果存入 Lo 寄存器。该指令支持 8 位的保护位。
- 影响标志位 :** 无影响
- 溢出标志位:** 有影响
- 指令格式 :**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	1	0		寄存器 Y		寄存器 X				

## MULT——乘法指令

- 操作：** RX  $\leftarrow$  RX  $\times$  RY  
**语法：** mult rx, ry  
**说明：** 寄存器 RX 的值乘以寄存器 RY 的值，并把结果的低 32 位存入寄存器 RX 中。32 $\times$ 32 ->32(低位)。不管源操作数被认为是有符号数还是无符号数，结果都是一样的。  
**影响标志位：** 无影响。  
**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	1		寄存器 Y		寄存器 X				

## MULU<sup>610E</sup>——无符号 32 位乘法指令

- 操作：**  $(u)\{Hi[31:0] | Lo[31:0]\} \leftarrow (u)Rx[31:0] \times (u)Ry[31:0]$
- 语法：** mulu rx , ry
- 说明：** 寄存器 X 的值和寄存器 Y 的值无符号相乘。积的低 32 位存入 Lo 寄存器，高 32 位存入 Hi 寄存器。
- 影响标志位：** 无影响。
- 溢出标志位：** 溢出位清零
- 指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	1	0	0		寄存器 Y		寄存器 X				

## MULUA<sup>610E</sup>——无符号 32 位乘累加

**操作：**  $(u)\{Hi[31:0] | Lo[31:0]\} \leftarrow (u)\{Hi[31:0] | Lo[31:0]\} + (u)Rx[31:0] \times$

$(u)Ry[31:0]$

**语法：**

mulua rx , ry

**说明：**

寄存器 X 的值和寄存器 Y 的值无符号相乘，相乘的积和 {Hi[31:0]|Lo[31:0]} 相加。最终结果的低 32 位存入 Lo 寄存器，高 32 位存入 Hi 寄存器。该指令不支持保护位。

**影响标志位：** 无影响。

**溢出标志位：** 有影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	1	0	1		寄存器 Y		寄存器 X				

## MULUS<sup>610E</sup>——无符号 32 位乘累减指令

**操作：**  $(u)\{Hi[31:0] | Lo[31:0]\} \leftarrow (u)\{Hi[31:0] | Lo[31:0]\} - (u)Rx[31:0] \times$

$(u)Ry[31:0]$

**语法：**

mulus rx , ry

**说明：**

寄存器 X 的值和寄存器 Y 的值无符号相乘，然后  $\{Hi[31:0]|Lo[31:0]\}$  减去相乘的积。最终结果的低 32 位存入 Lo 寄存器，高 32 位存入 Hi 寄存器。该指令不支持保护位。

**影响标志位：** 无影响。

**溢出标志位：** 有影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	1	1	0		寄存器 Y		寄存器 X				

## MVC——位 C 传送指令

**操作：** RX  $\leftarrow$  C

**语法：** mvc rx

**说明：** 把位 C 的值复制到目标寄存器 RX 的最低位中，RX 其他的位清零。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	1	0				寄存器 X

## MVCV——位 C 取反传送指令

**操作：**  $RX \leftarrow (!C)$

**语法：** mvcv rx

**说明：** 把位 C 取反后复制到目标寄存器 RX 的最低位中，RX 其他的位清零。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	1	1				寄存器 X

## MVTC<sup>610E</sup>——溢出位复制到 C 位指令

**操作：**  $C \leftarrow V$

**语法：** mvtc

**说明：** 将 DCSR(CR14)的溢出位复制到 PSR(CR0)的 C 位。

**影响标志位：** 有影响

**溢出标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

## **NOT—按位非指令**

**操作：**  $\text{RX} \leftarrow (\text{!RX})$

**语法：** not rx

**说明：** 寄存器 RX 按位取反。

**影响标志位：**无影响

### **指令格式：**

## OMFLIP<sup>620</sup>——基于 omega-flip 网络的总体置換指令

**操作：**  $(HI, LO) \leftarrow \text{omega}(HI, LO, c, rx, ry)$

**语法：** omega c, rx, ry

**说明：** 此指令定义了两级映射，每一级可以被配置成 omega 网络，也可以被配置成 flip 网络（由 C 控制）。寄存器 RX 和 RY 被认为是用来控制置換的配置。寄存器 X 控制第一级的映射，寄存器 Y 控制第二级的映射。寄存器的每一位控制冲突双方的路由通路，0 代表不能通过，1 代表能通过。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	1	0	0	0	0	C									寄存器 Y	寄存器 X

C——网络类型。

00——omega-omega

01——omega-flip

10——flip-omega

11——flip-flip

## OR——按位或指令

**操作：** RX  $\leftarrow$  RX+RY

**语法：** or rx, ry

**说明：** 寄存器 RX 和寄存器 RY 进行按位或操作，并把结果存在 RX 中。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	1	0		寄存器 Y		寄存器 X				

## PSRCLR——PSR 清零指令

**操作：** PSR({AF, FE, IE})  $\leftarrow$  0

**语法：** psrclr af;  
psrclr ee;  
psrclr fe;  
psrclr ie;  
psrclr ee, fe;  
psrclr ee, ie;  
psrclr fe, ie;  
psrclr ee, fe, ie.

**属性：** 有较高的优先级。

**说明：** 选中的 PSR 位被清零。

**限制：** 能被同时选中位的组合是受限制的。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	1	1	1	1	1	0		位选	

位选域——指定要被清零的位。

000 —— PSR(AF)

xx1 —— PSR(IE)

x1x —— PSR(FE)

1xx —— PSR(EE)

## PSRSET——PSR 置位指令

**操作：** PSR({AF, FE, IE})  $\leftarrow$  1

**语法：**

```
psrset af;  
psrset ee;  
psrset fe;  
psrset ie;  
psrset ee, fe;  
psrset ee, ie;  
psrset fe, ie;  
psrset ee, fe, ie.
```

**属性：** 有较高的优先级。

**说明：** 选中的 PSR 位被置 1。

**限制：** 能被同时选中位的组合是受限制的。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	位选
0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	

位选域——指定要被清零的位。

000 —— PSR(AF)

xx1 —— PSR(IE)

x1x —— PSR(FE)

1xx —— PSR(EE)

## RFI——快速中断返回指令

<b>操作：</b>	PC $\leftarrow$ FPC, PSR $\leftarrow$ FPSR
<b>语法：</b>	rfi
<b>属性：</b>	有较高的优先级。
<b>说明：</b>	PC 值为存在于控制寄存器 FPC 中的值, PSR 值为存在于 FPSR 的值, 指令执行从新的 PC 地址处开始。
<b>影响标志位：</b>	无影响
<b>指令格式：</b>	

## ROTLI——立即数循环左移指令

**操作：**  $RX \leftarrow RX$  循环左移 IMM5 位

**语法：** rotli rx, imm5

**说明：** 寄存器 RX 的值循环左移，左移的位数由 IMM5 决定。

**限制：** 立即数的范围为 1-31。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	0	0			IMM5						寄存器 X

IMM5 域——指定立即数的值。

00001——1

00010——2

.....

11111——31

## RSUB——减法指令

**操作：**  $RX \leftarrow RY - RX$

**语法：** rsub rx, ry

**说明：** 寄存器 Y 的值减去寄存器 RX 的值，并把结果存在寄存器 X 中。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	0	0		寄存器 Y		寄存器 X				

## RSUBI——立即数减法指令

**操作：**

$RX \leftarrow \text{无符号IMM5-RX}$

**语法：**

rsubi rx, imm5

**说明：**

无符号 IMM5 的值减去寄存器 RX 的值，并把结果存在寄存器 X 中。

**限制：**

立即数的范围为 0-31。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	0	0		IMM5							寄存器 X

IMM5 域——指定立即数的值。

00000——0

00001——1

.....

11111——31

## RTE——异常情况返回指令

<b>操作</b>	PC $\leftarrow$ EPC, PSR $\leftarrow$ EPSR
<b>语法</b>	rte
<b>属性</b>	有较高的优先权。
<b>说明</b>	PC 值为存在于控制寄存器 CR4 的值, PSR 为存在于 CR2 的值, 指令执行从新的 PC 值指令开始。
<b>影响标志位</b>	无影响
<b>指令格式</b>	

## SEXTB——字节符号扩展指令

**操作：**  $RX \leftarrow RX[7:0]$  符号位扩展到32位

**语法：** sextb rx

**说明：** 符号扩展寄存器的最低字节到 32 位。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	寄存器 X

## SEXTH——半字符号扩展指令

**操作：**  $RX \leftarrow RX[15:0]$  符号位扩展到32位

**语法：** sexth rx

**说明：** 符号扩展寄存器的低半字到 32 位。

**影响标志位：** 无影响。

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	寄存器 X

## ST.[B,H,W]——从寄存器存入存储器

**操作：** 存储器  $\leftarrow$  源寄存器

MEM[RX + unsigned IMM4<<\{0,1,2\}]  $\leftarrow$  RZ

**语法：** st.[b,h,w] rz, (rx, disp)

**说明：** 存储寄存器的内容到存储器。存储操作有三个选项：w(字)，h(半字)和 b (字节)。通过 4 位立即数，存储选项的移位，并进行零拓展来获得偏移量。然后把偏移量与寄存器 RX 的值相加等到新的地址。最后把 RZ 的值存放到新的地址。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	SIZE	1		寄存器 Z		IMM4		寄存器 X							

SIZE——说明装入数据单位的大小。

00——字

01——字节

10——半字

IMM4 域——指定 4 位立即数。

## STM——多个寄存器存储到存储器指令

- 操作：** 存储器  $\leftarrow$  源寄存器  
**语法：** stm rf-r15, (r0)  
**说明：** 多个寄存器存储到存储器。stm 指令用于从一些邻近的寄存器向堆栈存储数据。寄存器 R0 将作为基址指针，寄存器 Rf-R15 依次向存储器存储数据。  
**限制：** RF 只能是 R1—R14。  
**影响标志位：** 无影响  
**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	1	1	1				寄存器 F

**STOP——进入低功耗暂停模式指令**

**操作：** 进入低功耗暂停模式

**语法：** stop

**属性：** 较高的优先级

**说明：**此指令使处理器进入低功耗模式，并等待一个中断来退出这个模式。此时，CPU 时钟停止，大部分外围设备也被停止。

**影响标志位：**无影响

### **指令格式：**

## STQ——4个寄存器存入存储器指令

**操作：**

存储器  $\leftarrow$  源寄存器

**语法：**

stq r4-r7, (rx)

**说明：**

stq 指令用于从 4 个寄存器 (R4—R7) 向存储器存储数据。寄存器 RX 指定第一个被存储数据的位置。

**限制：**

RX 不能为 R4—R7。

**影响标志位：**

无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	1	0	1		寄存器 X		

## SUBC——无符号带进位减法指令

**操作：**  $RX \leftarrow RX - RY - (!C)$  ,  $C \leftarrow$  进位

**语法：** subc rx, ry

**说明：** 寄存器 X 的值减去寄存器 RY 的值和位 C 的非值，并把结果存在寄存器 X 中。位 C 将会因为进位而更新，对于减法来说，位 C 的值为借位取反。

**影响标志位：**  $C \leftarrow$  进位

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	1		寄存器 Y		寄存器 X				

## SUBI——无符号立即数减法指令

**操作：**  $RX \leftarrow RX - \text{无符号OIMM5}$

**语法：** subi rx, oimm5

**说明：** 寄存器 X 的值减去立即数的值。

**限制：** OIMM5 的范围在 1-32 之间。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	1	0		OIMM5							寄存器 X

OIMM5 域——指定立即数的值。

注意：二进制数比实际值会偏移 1。

00000——减 1

00001——减 2

.....

11111——减 32

## SUBU——无符号减法指令

**操作：**  $RX \leftarrow RX - RY$

**语法：** subu rx, ry

sub rx, ry

**说明：** 寄存器 X 的值减去寄存器 Y 值，并把结果存在寄存器 X 中。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	1		寄存器 Y		寄存器 X				

## SYNC——CPU 同步指令

**操作：**使CPU同步

**语法：** sync

**说明：**当处理器碰到 sync 指令时，指令就会被悬挂起来直到所有外面的操作全都完成，即没有未完成的指令。

**影响标志位：**无影响

## 指令格式：

## TRAP——无条件操作系统陷阱指令

**操作：**引起陷阱异常发生

**语法：**trap #imm2

**说明：**当处理器碰到 trap 指令时，发生陷阱异常操作。

**影响标志位：**无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	IMM2	

IMM2——2 位立即数域用来指定陷阱数。

## TST——零测试指令

**操作：** if( $RX \& RY \neq 0$ ), then  
     $C \leftarrow 1$  ;

    else  
         $C \leftarrow 0$ 。

**语法：** tst rx, ry

**说明：** 测试寄存器 X 和寄存器 Y 逻辑与的结果。

如果结果是非 0，则位 C 为 1；否则，位 C 为 0。

**影响标志位：** if( $RX \& RY \neq 0$ ), 位 C 为 1；否则，位 C 为 0。

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0		寄存器 Y		寄存器 X				

## TSTNBZ——无字节等于零寄存器测试指令

**操作：** 如果没有字节等于零, then

C ← 1 ;

else

C ← 0。

**语法：** tstnbz rx

**说明：** 测试寄存器 X 中是否没有字节等于零。如果结果是真（即没有字节等于零），则位 C 为 1；否则，位 C 为 0。

**影响标志位：** 如果寄存器 X 中没有等于零的字节，则位 C 为 1，否则，位 C 为 0。

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	1	0	0	1		寄存器 X		

## VMULSH<sup>610E</sup>——两路有符号 16 位乘法指令

- 操作：**  $(s)Hi[31:0] \leftarrow (s)Rx[31:16] \times (s)Ry[31:16]$   
 $(s)Lo[31:0] \leftarrow (s)Rx[15:0] \times (s)Ry[15:0]$
- 语法：** vmulsh rx , ry
- 说明：** 寄存器 X 的高 16 位和寄存器 Y 的高 16 位有符号相乘，32 位的积存入 Hi 寄存器。寄存器 X 的低 16 位和寄存器 Y 的低 16 位有符号相乘，32 位的积存入 Lo 寄存器。
- 影响标志位：** 无影响
- 溢出标志位：** 溢出位清零
- 指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	0	0	0		寄存器 Y		寄存器 X				

## VMULSHA<sup>610E</sup>——两路有符号 16 位乘累加指令

**操作 :**  $(s)\{Hi[31:0]\} \leftarrow (s)\{Hi[31:0]\} + (s)Rx[31:16] \times (s)Ry[31:16]$

$(s)\{Lo[31:0]\} \leftarrow (s)\{Lo[31:0]\} + (s)Rx[15:0] \times (s)Ry[15:0]$

**语法 :** vmulsha rx , ry

**说明 :** 寄存器 X 的高 16 位和寄存器 Y 的高 16 位有符号相乘，32 位的积和 Hi 寄存器的值相加，最终结果存入 Hi 寄存器。寄存器 X 的低 16 位和寄存器 Y 的低 16 位有符号相乘，32 位的积和 Lo 寄存器的值相加，最终结果存入 Lo 寄存器。指令的每一路都支持 8 位的保护位。任何一路溢出都会使溢出位置 1。

**影响标志位 :** 无影响

**溢出标志位:** 有影响

**指令格式 :**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	0	0	1		寄存器 Y		寄存器 X				

## VMULSHS<sup>610E</sup>——两路有符号 16 位乘累减指令

**操作 :**  $(s)\{Hi[31:0]\} \leftarrow (s)\{Hi[31:0]\} - (s)Rx[31:16] \times (s)Ry[31:16]$

$(s)\{Lo[31:0]\} \leftarrow (s)\{Lo[31:0]\} - (s)Rx[15:0] \times (s)Ry[15:0]$

**语法 :** vmulshs rx , ry

**说明 :** 寄存器 X 的高 16 位和寄存器 Y 的高 16 位有符号相乘，然后 Hi 寄存器的值减去 32 位的积，最终结果存入 Hi 寄存器。寄存器 X 的低 16 位和寄存器 Y 的低 16 位有符号相乘，Lo 寄存器的值减去 32 位的积，最终结果存入 Lo 寄存器。指令的每一路都支持 8 位的保护位。任何一路溢出都会使溢出位置 1。

**影响标志位 :** 无影响

**溢出标志位:** 有影响

**指令格式 :**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	0	1	0		寄存器 Y		寄存器 X				

## VMULSW<sup>610E</sup>——两路有符号 16x32 乘法指令

**操作：**  $(s)Hi[31:0] \leftarrow ((s)Rx[31:16] \times (s)Ry[31:0])[47:16]$

$(s)Lo[31:0] \leftarrow ((s)Rx[15:0] \times (s)Ry[31:0])[47:16]$

**语法：** vmulsw rx , ry

**说明：** 寄存器 X 的高 16 位和寄存器 Y 的值有符号相乘，48 位积的高 32 位存入 Hi 寄存器。寄存器 X 的低 16 位和寄存器 Y 的值有符号相乘，48 位积的高 32 位存入 Lo 寄存器。

**影响标志位：** 无影响

**溢出标志位：** 溢出位清零

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	1	0	0		寄存器 Y		寄存器 X				

## VMULSWA<sup>610E</sup>——两路有符号 16x32 乘累加指令

**操作 :**  $(s)\{Hi[31:0]\} \leftarrow (s)\{Hi[31:0]\} + ((s)Rx[31:16] \times (s)Ry[31:0])[47:16]$

$(s)\{Lo[31:0]\} \leftarrow (s)\{Lo[31:0]\} + ((s)Rx[15:0] \times (s)Ry[31:0])[47:16]$

**语法 :** vmulswa rx , ry

**说明 :** 寄存器 X 的高 16 位和寄存器 Y 的值有符号相乘，48 位积的高 32 位和 Hi 寄存器的值相加，最终结果存入 Hi 寄存器。寄存器 X 的低 16 位和寄存器 Y 的值有符号相乘，48 位积的高 32 位和 Lo 寄存器的值相加，最终结果存入 Lo 寄存器。该指令的每一路都支持 8 位的保护位。任何一路溢出都会导致溢出位置 1。

**影响标志位 :** 无影响

**溢出标志位:** 有影响

**指令格式 :**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	1	0	1		寄存器 Y		寄存器 X				

## VMULSWS<sup>610E</sup>——两路有符号 16x32 乘累减指令

**操作 :**  $(s)\{Hi[31:0]\} \leftarrow (s)\{Hi[31:0]\} - ((s)Rx[31:16] \times (s)Ry[31:0])[47:16]$

$(s)\{Lo[31:0]\} \leftarrow (s)\{Lo[31:0]\} - ((s)Rx[15:0] \times (s)Ry[31:0])[47:16]$

**语法 :** vmuls ws rx , ry

**说明 :** 寄存器 X 的高 16 位和寄存器 Y 的值有符号相乘，然后 Hi 寄存器的值减去积的高 32 位，最终结果存入 Hi 寄存器。寄存器 X 的低 16 位和寄存器 Y 的值有符号相乘，然后 Lo 寄存器的值减去积的高 32 位，最终结果存入 Lo 寄存器。该指令的每一路都支持 8 位的保护位。任何一路溢出都会导致溢出位置 1。

**影响标志位 :** 无影响

**溢出标志位:** 有影响

**指令格式 :**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	1	1	0		寄存器 Y		寄存器 X				

**WAIT——停止执行等待中断指令**

<b>操作：</b>	进入低功耗等待模式
<b>语法：</b>	wait
<b>属性：</b>	有较高的优先权。
<b>说明：</b>	此指令停止当前指令执行，并等待一个中断，此时CPU时钟停止。所有的外围设备都仍在继续运行，并有可能会产生中断而引起CPU从等待模式退出。
<b>影响标志位：</b>	无影响
<b>指令格式：</b>	

## XOR——按位异或指令

**操作：**  $RX \leftarrow RX \wedge RY$

**语法：** xor rx, ry

**说明：** 寄存器 RX 和寄存器 RY 进行按位异或操作，并把结果存在寄存器 RX。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	1	1		寄存器 Y		寄存器 X				

## XSR——扩展右移指令

**操作：** 寄存器带位C循环右移一位。

**语法：** xsr rx

**说明：** 带位 C 寄存器 RX 循环右移 1 位：

Ctmp $\leftarrow$ C ;

C←RX[0] ;

lsr(RX, 1) ;

RX[31] ← Ctmp.

**影响标志位：** C $\leftarrow$ RX[0]

## 指令格式：

## XTRB0——提取字节 0 到 R1 并进行零扩展指令

**操作：**  $R1 \leftarrow$  寄存器的字节 0 (位 31:24) 零扩展到 32 位

**语法：** xtrb0 r1, rx

**说明：** 提取寄存器字节 0 (位 31:24) 到 R1，并进行零扩展：

$R1[7:0] \leftarrow RX[31:24]$  ;

$R1[31:8] \leftarrow 0$  ;

if (result == 0), then

$C \leftarrow 0$  ;

else

$C \leftarrow 1$ 。

**影响标志位：** 如果结果不等于 0 则位 C 置 1，否则清零。

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	0	1	1				寄存器 X

## XTRB1——提取字节 1 到 R1 并进行零扩展指令

**操作：**  $R1 \leftarrow$  寄存器的字节1（位23:16）零扩展到32位

**语法：** xtrb1 r1, rx

**说明：** 提取寄存器字节 1（位 23:16）到 R1，并进行零扩展：

$R1[7:0] \leftarrow RX[23:16]$  ;

$R1[31:8] \leftarrow 0$  ;

if (result == 0), then

$C \leftarrow 0$  ;

else

$C \leftarrow 1$ 。

**影响标志位：** 如果结果不等于 0 则位 C 置 1，否则清零。

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	0	1	0				寄存器 X

## XTRB2——提取字节 2 到 R1 并进行零扩展指令

**操作：**  $R1 \leftarrow$  寄存器的字节2（位15:8）零扩展到32位

**语法：** xtrb2 r1, rx

**说明：** 提取寄存器字节 2（位 15:8）到 R1，并进行零扩展：

$R1[7:0] \leftarrow RX[15:8]$ ；

$R1[31:8] \leftarrow 0$ ；

if (result == 0), then

$C \leftarrow 0$ ；

else

$C \leftarrow 1$ 。

**影响标志位：** 如果结果不等于 0 则位 C 置 1，否则清零。

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	0	0	1				寄存器 X

## XTRB3——提取字节 3 到 R1 并进行零扩展指令

**操作：**  $R1 \leftarrow$  寄存器的字节3（位7:0） 零扩展到32位

**语法：** xtrb3 r1, rx

**说明：** 提取寄存器字节 3（位 7:0）到 R1，并进行零扩展:

$R1[7:0] \leftarrow RX[7:0]$  ;

$R1[31:8] \leftarrow 0$  ;

if (result == 0), then

$C \leftarrow 0$  ;

else

$C \leftarrow 1$ 。

**影响标志位：** 如果结果不等于 0 则位 C 置 1，否则清零。

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	寄存器 X

## ZEXTB——字节零扩展指令

**操作：** RX  $\leftarrow$  寄存器RX的低字节（位7:0）零扩展到32位

**语法：** zextb rx

**说明：** 寄存器 RX 的低字节零扩展到 32 位。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	寄存器 X

## ZEXTH——半字零扩展指令

**操作：** RX  $\leftarrow$  寄存器RX的低半字（位15:0）零扩展到32位

**语法：** zexth rx

**说明：** 寄存器 RX 的低半字零扩展到 32 位。

**影响标志位：** 无影响

**指令格式：**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	1	1	0		寄存器 X		