# 附錄一 ARC EM處理器功能模組

## 第一節 計時器功能

### 一、概述

ARC EM處理器包含以下計時器功能：

1. 兩個32位寬度的可程式設計計時器Timer0和Timer1；
2. 一個64位寬度的即時計數器RTC （Real-Time Counter）。

它們屬於可配置範疇，通過以下選項在系統組態時控制是否產生相應的硬體邏輯。

表 3： ARC EM處理器計時器配置選項清單

|  |  |  |  |
| --- | --- | --- | --- |
| **選項** | **取值範圍** | **預設值** | **描述** |
| HAS\_TIMER0 | 0,1 | 0 | 是否配置Timer0硬體邏輯0：不配置；1：配置 |
| HAS\_TIMER1 | 0,1 | 0 | 是否配置Timer1硬體邏輯0：不配置；1：配置 |
| RTC\_OPTION | 0,1 | 0 | 是否配置64位元RTC計數器0：不配置；1：配置 |

### 二、Starter Kit開發板計時器配置資訊

在ARC EM Starter Kit開發板中共包含四種EM處理器的映射邏輯：ARC EM4/ARC EM4\_16CR/ARC EM6/ARC EM6 GP。各計時器在四個映射中的配置細節如下表所示：

表 4：ARC EM Starter Kit計時器功能映射表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **映射****配置資訊** | **ARC EM4** | **ARC EM4\_16CR** | **ARC EM6** | **ARC EM6 GP** |
| **HAS\_TIMER0** | 1 | 1 | 1 | 1 |
| **HAS\_TIMER1** | 1 | 0 | 1 | 0 |
| **RTC\_OPTION** | 0 | 0 | 0 | 0 |

具體的計時器配置資訊也可以通過訪問不同映射邏輯下EM處理器的Timer BCR獲得。

表 5： ARC EM處理器計時器配置寄存器TIMER\_BUILD清單

|  |  |  |  |
| --- | --- | --- | --- |
| **Aux寄存器位址** | **輔助寄存器名** | **存取權限** | **描述** |
| 0x75 | TIMER\_BUILD | R | 計時器配置寄存器 |

TIMER\_BUILD寄存器的位定義佈局如下圖所示，

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **RTC** | **T1** | **T0** | **Version** |

圖 46： TIMER\_BUILD寄存器佈局圖

表 6： TIMER\_BUILD寄存器位域定義列表

|  |  |
| --- | --- |
| **位域** | **描述** |
| Version | 版本資訊。0x04，代表基於ARCv2 ISA架構的處理器系列 |
| T0 | 是否有Timer00：無1：有 |
| T1 | 是否有Timer00：無1：有 |
| RTC | 是否有RTC0：無1：有 |

### 三、Timer0相關輔助寄存器說明及使用

在ARC EM處理器體系結構中，與Timer0相關的輔助寄存器如下：

表 7： ARC EM處理器Timer0相關輔助寄存器列表

|  |  |  |  |
| --- | --- | --- | --- |
| **Aux寄存器位址** | **輔助寄存器名** | **存取權限** | **描述** |
| 0x21 | COUNT0 | RW | Timer0計時器計數值 |
| 0x22 | CONTROL0 | RW | Timer0計時器控制寄存器值 |
| 0x23 | LIMIT0 | RW | Timer0寄存器限制值 |

1. COUNT0

Timer0的COUNT0寄存器各位域佈局如下圖所示，

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Timer0計時器計數值** |

圖 47： COUNT0寄存器佈局圖

寫操作：對該寄存器執行寫操作用於設置Timer0的初始值。任何時刻寫入，都將導致計數器從該設定值開始遞增。

讀操作：對該寄存器執行讀操作將即時給出當前Timer0的計數值。

1. CONTROL0

Timer0的CONTROL0寄存器各位域佈局如下圖所示，

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|  | **IP** | **W** | **NH** | **IE** |

圖 48： CONTROL0寄存器佈局圖

CONTROL0寄存器主要起控制作用，用於控制Timer0的執行方式。

表 8： CONTROL0寄存器位域定義列表

|  |  |
| --- | --- |
| **位域** | **描述** |
| IE | Interrupt Enable。用於控制當Timer0的計數值與LIMIT0寄存器中設定值相等時，是否觸發中斷（Interrupt）。0：不觸發1：觸發系統重定後，該位元默認為0. |
| NH | Not Halted。控制處理器在不同狀態時Timer0的計數情況。0：只要供電，在任何工作模式下Timer0計數器每時鐘週期遞增1；1：只有當處理器出於執行狀態下（non-halted）時，Timer0計數器每時鐘週期遞增1系統重定後，該位元默認為0. |
| W | Watchdog模式位元。用於控制當Timer0計數值與LIMIT0寄存器中設定值相等時，是否觸發一個看門口重定（watchdog reset）硬體行為。0：不觸發1：觸發系統重定後，該位元默認為0. |
| IP | Interrupt Pending。該位用於指示當前是否有一個未處理的（pending）計時器中斷。 |

使用說明：

* 當IE和W位同時被置1時，只有W位設置值有效。因為當計數器值等於LIMIT0中設定值時，將觸發watchdog reset，此時所有中斷資訊也將被重定；
* 對CONTROL0寄存器各比特位的控制應當在同一個寫操作中完成；
* IP是一個狀態指示位元。當一個Timer0觸發的中斷未被回應之前，讀CONTROL0寄存器的IP位元顯示為1，當中斷被回應之後，IP位元顯示為0.一般而言，IP位的清除操作是在Timer0相應的中斷服務副程式中進行的。在該中斷服務副程式中，會首先讀取CONTROL0，執行遮罩操作遮罩掉IP位的值，再將處理後的值寫入CONTROL0寄存器。
1. LIMIT0

Timer0的LIMIT0寄存器各位域佈局如下圖所示，

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Timer0計時器限定值** |

圖 49： LIMIT0寄存器佈局圖

寫操作：對該寄存器執行寫操作將設置觸發中斷或者復位的一個限定值。

讀操作：對該寄存器執行讀操作將返回之前設置的Timer0計數限定值。

上電復位後，該寄存器的預設值為0x00FFFFFF。

### 四、Timer1相關輔助寄存器說明及使用

在ARC EM處理器體系結構中，與Timer1相關的輔助寄存器如下：

表 9： ARC EM處理器Timer1相關輔助寄存器列表

|  |  |  |  |
| --- | --- | --- | --- |
| **Aux寄存器位址** | **輔助寄存器名** | **存取權限** | **描述** |
| 0x100 | COUNT1 | RW | Timer1計時器計數值 |
| 0x101 | CONTROL1 | RW | Timer1計時器控制寄存器值 |
| 0x102 | LIMIT1 | RW | Timer1寄存器限制值 |

Timer1計時器的各輔助寄存器功能與位定義與Timer0一致，具體可參考前文內容。

## 第二節 中斷處理功能

### 一、概述

基於ARCv2 ISA體系結構的EM處理器中斷系統可支援多達256個中斷，其中中斷向量的0~15分配給處理器內部異常事件（Exceptions），中斷向量的16~255分配給處理器中斷（Interrupts）。

中斷系統具有良好的可程式設計性，具體包括：

1. 支援多達240個中斷，每個中斷可被定義為電平觸發（level sensitive）或脈衝觸發（pulse sensitive）；
2. 支持多達16個中斷優先順序。
3. 根據應用的需求，每個中斷的回應優先順序可被定義為0~15中的任意值，其中0代表最高優先順序，15代表最低優先順序；
4. 支援低優先順序的中斷處理被高優先順序的中斷處理打斷；
5. 根據不同中斷入口，自動進行上下文保存；
6. 軟體可配置STATUS32中的E域來設置中斷閾值，控制處理器僅對閾值之上的中斷進行回應或被喚醒。

### 二、中斷類型

 ARC EM處理器支援以下類型的中斷：

1. 計時器中斷：在配置了相應計時器的CONTROL寄存器之後，由Timer0或者Timer1在計數到達設定值（LIMITn）時所觸發的中斷。
2. 外部中斷：由外部系統觸發的中斷。
3. 軟體觸發中斷：由軟體觸發的中斷，類比外部硬體行為，可用於調試用途。

### 三、中斷向量表

 下表為EM處理器中各中斷向量的映射關係表。由Timer0觸發的中斷固定接中斷向量值16上；由Timer1觸發的中斷固定接中斷向量值17上。外部中斷序號動態分配，如當硬體中無Timer0和Timer1時，中斷序號可從16~240任意分配；當Timer1使能時，外部中斷的序號可分配在16、18~240區間。

表 10： ARC EM處理器中斷向量表

|  |  |  |  |
| --- | --- | --- | --- |
| **中斷向量值****（Interrupt Vector）** | **描述** | **中斷向量值****（Interrupt Vector）** | **描述** |
| 0 | Reset | 15 | Unused |
| 1 | Memory Error | 16 | Timer0/ext int |
| 2 | Instruction Error | 17 | Timer1/ext int |
| 3 | EV\_MachineCheck | 18 | ext int |
| 4 | EV\_TLBMissI | … |  |
| 5 | EV\_TLBMissD | 41 | ext int |
| 6 | EV\_ProtV | 42 | ext int |
| 7 | EV\_PrivilegeV | 43 | ext int |
| 8 | EV\_SWI | … |  |
| 9 | EV\_Trap | 150 | ext int |
| 10 | EV\_Extension | 151 | ext int |
| 11 | EV\_DivZero | … |  |
| 12 | EV\_DCError | 253 | ext int |
| 13 | EV\_Malligned | 254 | ext int |
| 14 | Unused | 255 | ext int |

### 四、中斷向量基底位址

當某個異常或中斷發生時，處理器根據中斷向量基底位址（Interrupt Vector Base）加上當前中斷的向量偏移值（Interrupt Vector）計算中斷入口的位址，從而進入相應的中斷服務副程式。

其中，中斷向量基底位址（Interrupt Vector Base）通過輔助寄存器INT\_VECTOR\_BASE進行設置。

表 11： ARC EM處理器中斷向量基底位址INT\_VECTOR\_BASE列表

|  |  |  |  |
| --- | --- | --- | --- |
| **Aux寄存器位址** | **輔助寄存器名** | **存取權限** | **描述** |
| 0x25 | INT\_VECTOR\_BASE | RW | 中斷向量基底位址寄存器 |

中斷向量基底位址以1KB對齊，對其低10位進行任何寫操作將會被忽略。而中斷向量基底位址的有效位元域取決於硬體系統組態時的PC定址空間。在PC定址空間為4GB以及128MB時，INT\_VECTOR\_BASE寄存器各位域佈局分別如下圖所示，

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **INT\_VECTOR\_BASE[31:10]** |  |

圖 50： INT\_VECTOR\_BASE佈局圖（PC\_SIZE=32）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|  | **INT\_VECTOR\_BASE[PC\_SIZE-1:10]** |  |

圖 51：ＩＮＴ\_ＶＥＣＴＯＲ\_ＢＡＳＥ佈局圖（ＰＣ\_ＳＩＺＥ＝２８）

### 五、中斷程式設計相關輔助寄存器

下表列出了程式設計中常用的中斷相關輔助寄存器。

表 12： ARC EM處理器中斷程式設計相關輔助寄存器列表

|  |  |  |  |
| --- | --- | --- | --- |
| **Aux寄存器位址** | **輔助寄存器名** | **存取權限** | **描述** |
| 0xF3 | IRQ\_BUILD | R | 中斷系統組態寄存器 |
| 0x0E | AUX\_IRQ\_CTRL | RW | 中斷上下文保存控制寄存器 |
| 0x43 | AUX\_IRQ\_ACT | RW | 活躍中斷（Active Interrupt）寄存器 |
| 0x40A | ICAUSE | R | 中斷原因寄存器 |
| 0x40B | IRQ\_SELECT | RW | 中斷選擇寄存器 |
| 0x40C | IRQ\_ENABLE | RW | 中斷使能寄存器 |
| 0x40D | IRQ\_TRIGGER | RW | 中斷觸發模式寄存器 |
| 0x40F | IRQ\_STATUS | R | 中斷狀態寄存器 |
| 0x415 | IRQ\_PULSE\_CANCLE | W | 脈衝觸發中斷清除寄存器 |
| 0x416 | IRQ\_PENDING | R | 未回應中斷狀態寄存器 |
| 0x200 | IRQ\_PRIORITY\_PENDING | R | 未回應中斷優先順序狀態寄存器 |
| 0x206 | IRQ\_PRIORITY | RW | 中斷優先順序配置寄存器 |

1. IRQ\_BUILD

IRQ\_BUILR寄存器屬於BCR寄存器範疇，它只具有唯讀功能，用於使用者通過程式訪問處理器硬體獲知處理器中斷系統資訊。

IRQ\_BUILD寄存器的各位域佈局如下圖所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **RAZ** | **F** | **P[3:0]** | **EXTS[7:0]** | **IRQS[7:0]** | **Version** |

圖 52：IRQ\_BUILD寄存器佈局圖

表 13：IRQ\_BUILD寄存器位域定義表

|  |  |
| --- | --- |
| **位域** | **描述** |
| Version | EM處理器中斷單元版本資訊0x01=ARCv2 |
| IRQS[7：0] | 指示EM處理器的中斷系統組態了多少個中斷 |
| EXTS[7：0] | 指示EM處理器的中斷系統組態了多少個外部中斷 |
| P[3：0] | 指示EM處理器的中斷系統組態了多少個回應中斷級（N-1） |
| F | 是否開啟快速中斷回應（FIRQ\_OPTION） |

1. AUX\_IRQ\_CTRL

AUX\_IRQ\_CTRL寄存器用於控制當中斷發生，進入或跳出中斷服務副程式時上下文的保存行為。上電復位後，該寄存器的初始值為0x00000000.

AUX\_IRQ\_CTRL寄存器的各位域佈局如下圖所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **RAZ/IOW** | **LP** | **R** | **U** | **L** | **B** | **RAZ/IOW** | **NR[4：0]** |

圖 53： AUX\_IRQ\_CTRL寄存器佈局圖

表 14： AUX\_IRQ\_CTRL寄存器位域定義表

|  |  |
| --- | --- |
| **位域** | **描述** |
| NR[4：0] | 指示用於中斷上下文保存的通用寄存器對數，值域為0~16.REG\_NUM\_REGS = 32， NR[4：0]最大為16REG\_NUM\_REGS = 16， NR[4：0]最大為8 |
| B | 指示是否要保存和重新載入BLINK寄存器值 |
| L | 指示是否要保存和重新載入ZOL相關寄存器（LP\_COUNT, LP\_START, LP\_END） |
| U | 指示在使用者模式下發生的中斷是否要保存到使用者堆疊（user stack） |
| R | 保留位 |
| LP | 指示是否要保存和重新載入代碼密度（code\_density）相關寄存器（EI\_BASE, JLI\_BASE, LDI\_BASE） |

1. AUX\_IRQ\_ACT

AUX\_IRQ\_ACT寄存器指示當前回應的中斷處於何種回應優先順序。每個優先順序對應一個活躍位（Active）。

AUX\_IRQ\_ACT寄存器的各位域佈局如下圖所示，

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **U** |  | **Active[15:0]** |

圖 54：AUX\_IRQ\_ACT寄存器佈局圖

表 15：AUX\_IRQ\_ACT寄存器位域定義表

|  |  |
| --- | --- |
| **位域** | **描述** |
| Active[15:0] | 中斷活躍位。每位對應一個中斷響應優先順序。Priority0對應Active[0]，priority15對應Active[15]。如果系統中沒有配置所有的中斷優先順序，則未配置優先順序對應位元的讀寫操作無效。 |
| U | 在Active[15:0]==0時保存STATUS32中U域值，記錄最外層中斷嵌套的工作模式（使用者模式/核心模式） |

上電復位後，該寄存器初始值為0x00000000.

Tips：該寄存器一般用於讀操作，查詢當前回應的中斷優先順序。

1. ICAUSE

中斷原因寄存器ICAUSE是一個唯讀寄存器，它實際上映射到多個物理寄存器上，每個中斷優先順序對應一個特定的ICAUSE寄存器。

當AUX\_IRQ\_ACT寄存器的值為非零時，讀ICAUSE寄存器將返回AUX\_IRQ\_ACT中處於活躍狀態的最高優先順序所對應的中斷數。

當AUX\_IRQ\_ACT寄存器值等於零（即無中斷發生）時，讀ICAUSE寄存器返回一個不確定值。

例如，AUX\_IRQ\_ACT=32’b1時，代表Priority0優先順序處發生了中斷。讀取ICAUSE寄存器，假設ICAUSE=32’h3將指示在當前時刻優先順序為Priority0的中斷同時發生了3個。

ICAUSE寄存器的各位域佈局如下圖所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **ICAUSE[m-1:0]** |

圖 55：ICAUSE寄存器佈局圖

其中，m=ceil(log2(M + 16))，M代表系統組態時支持的中斷數（IRQ\_BUILD[IRQS]）

上電復位後，該寄存器的初始值為0x00000000.

Tips：該寄存器主要用於處理器中斷系統判斷和調度中斷回應的先後次序。

1. IRQ\_SELECT

中斷選擇寄存器用於選擇特定的中斷號。IRQ\_SELECT寄存器的各位域佈局如下圖所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **Interrupt[m-1:0]** |

圖 56：IRQ\_SELECT寄存器佈局圖

其中，m=ceil(log2(M + 16))，M代表系統組態時支持的中斷數（IRQ\_BUILD[IRQS]）

上電復位後，該寄存器的初始值為0x00000000.

**注意**：中斷向量的0~15分配給處理器異常，所以配置該寄存器的有效中斷號應從16開始。

1. IRQ\_ENABLE

IRQ\_ENABLE寄存器設置某個中斷（由IRQ\_SELECT選定）是否開啟或者關閉。

IRQ\_ENABLE寄存器的位域佈局如下圖所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **E** |

圖 57：IRQ\_ENABLE寄存器佈局圖

表 16：IRQ\_ENABLE寄存器位域定義表

|  |  |
| --- | --- |
| **位域** | **描述** |
| E | 中斷使能位。0：關閉（disable）1：開啟（enable） |

上電復位後，該寄存器的初始值為0x00000001.

1. IRQ\_TRIGGER

IRQ\_TRIGGER寄存器設置某個中斷（由IRQ\_SELECT選定）的觸發模式是電平觸發還是脈衝觸發。IRQ\_TRIGGER寄存器的位域佈局如下圖所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **E** |

圖 58： IRQ\_TRIGGER寄存器佈局圖

表 17：IRQ\_TRIGGER寄存器位域定義表

|  |  |
| --- | --- |
| **位域** | **描述** |
| T | 中斷觸發模式位元。0：電平觸發（level-sensitvie）1：脈衝觸發（pulse-sensitve） |

上電復位後，該寄存器的初始值為0x00000000.

1. IRQ\_STATUS

IRQ\_STATUS寄存器是一個唯讀寄存器，用於指示某個中斷（由IRQ\_SELECT選定）的所有資訊和狀態。

IRQ\_STATUS寄存器的各位域佈局如下圖所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **IP** | **Reserved** | **T** | **E** | **P[3：0]** |

圖 59：IRQ\_STATUS寄存器佈局圖

表 18：IRQ\_STATUS寄存器位域定義表

|  |  |
| --- | --- |
| **位域** | **描述** |
| P[3:0] | 由IRQ\_PRIORITY寄存器設定的中斷回應優先順序狀態位元 |
| E | 由IRQ\_ENABLE寄存器設定的中斷使能狀態位元 |
| T | 由IRQ\_TRIGGER寄存器設定的中斷觸發模式狀態位元 |
| IP | 由IRQ\_PENDING寄存器顯示的中斷回應狀態位元（pending） |

1. IRQ\_PULSE\_CANCLE

IRQ\_PULSE\_CANCLE寄存器是一個隻寫寄存器，用於從軟體方式清除某個脈衝觸發中斷（由IRQ\_SELECT選定）狀態。

IRQ\_PULSE\_CANCEL寄存器的各位域佈局如下圖所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **C** |

圖 60：IRQ\_PULSE\_CANCLE寄存器佈局圖

表 19： IRQ\_PULSE\_CANCEL寄存器位域定義表

|  |  |
| --- | --- |
| **位域** | **描述** |
| C | 中斷清除位。0：無效1：清除脈衝觸發的中斷 |

1. IRQ\_PENDING

IRQ\_PENDING寄存器為唯讀寄存器，用於指示某個中斷（由IRQ\_SELECT選定）是否處於未被回應（pending）狀態。

IRQ\_PENDING寄存器的各位域佈局如下圖所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **IP** |

圖 61： IRQ\_PENDING寄存器佈局圖

表 20： IRQ\_PENDING寄存器位域定義表

|  |  |
| --- | --- |
| **位域** | **描述** |
| IP | Interrupt Pending Status。0：無未回應中斷1：有未回應中斷 |

上電復位後，該寄存器的初始值為0x00000000.

Tips：該寄存器為程式提供某個中斷的回應狀態，一般可用於程式對中斷的程式設計調度。

1. IRQ\_PRIORITY\_PENDING

IRQ\_PRIORITY\_PENDING寄存器是一個唯讀寄存器，指示在當前正響應的中斷所處優先順序以下，所有未被處理的中斷所對應的優先順序向量值。每個優先順序對已一個標誌位元。

IRQ\_PRIORITY\_PENDING寄存器的各位域佈局如下圖所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **RAZ** | **Pending[n-1:0]** |

圖 62： IRQ\_PRIORITY\_PENDING寄存器佈局圖

表 21： IRQ\_PRIORITY\_PENDING寄存器位域定義表

|  |  |
| --- | --- |
| **位域** | **描述** |
| Pending[n-1:0] | 中斷未回應狀態位元。每位對應一個中斷響應優先順序。Priority0對應Pending[0]，priority15對應Pending[15]。如果系統中沒有配置所有的中斷優先順序，則未配置優先順序對應位元無效。0：該優先順序下無未回應的中斷；1：該優先順序下有未回應的中斷 |

上電復位後，該寄存器的初始值為0x00000000.

1. IRQ\_PRIORITY

IRQ\_PRIORITY寄存器設置某個中斷（由IRQ\_SELECT選定）的中斷回應優先順序，其各位域佈局如下圖所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **P[n-1:0]** |

圖 63： IRQ\_PRIORITY寄存器佈局圖

其中，n=ceil(log2(N))，N代表系統組態時處理器支持的中斷優先順序數（IRQ\_BUILD[P]）。

上電復位後，該寄存器的初始值為0x00000000.

### 六、中斷程式設計

ARC EM處理器的中斷系統支援對多個中斷進行回應（1≤M≤240），每個中斷的程式設計和控制通過一系列特定的輔助寄存器實現：

1. 配置IRQ\_SELECT寄存器選定中斷向量（e.g. int\_18）；
2. 配置IRQ\_PRIORITY寄存器設置int\_18的中斷回應優先順序；
3. 配置IRQ\_ENABLE寄存器，控制是否使能當前中斷。即，當int\_18發生時，處理器是否需要回應該中斷，還是忽略；
4. 配置IRQ\_TRIGGER寄存器，控制中斷的有效觸發方式是電平觸發還是脈衝觸發；
5. 配置IRQ\_PULSE\_CANCEL寄存器，控制是否需要清除一個未回應（pending）的脈衝觸發中斷；
6. 讀取IRQ\_PENDING寄存器的值，查看當前是否有未回應（pending）中斷
7. 讀取IRQ\_STATUS寄存器的值，指示由IRQ\_SELECT寄存器選定的中斷（e.g. int\_18）的配置和狀態資訊，包括是否使能，中斷回應優先順序，觸發模式以及是否有pending等。

對EM處理器中斷系統進行程式設計的過程如下：

1. 關閉（disable）所有中斷，並把當前已經發生但還未回應（pending）的中斷回應完成；關閉中斷可以通過CLRI命令實現，它將對STATUS32寄存器中的IE域置0。
2. 配置IRQ\_SELECT寄存器選定所需配置的中斷號
	1. 設置該中斷的回應優先順序（IRQ\_PRIORITY）。預設情況下，中斷的回應優先順序為0；
	2. 設置該中斷觸發模式（IRQ\_TRIGGER）是電平觸發還是脈衝觸發。預設情況下，中斷觸發模式為電平觸發；
	3. 設置該中斷是否使能（IRQ\_ENABLE）。
	4. 如果是需要撤銷一個脈衝觸發的未回應中斷，可以對IRQ\_PULSE\_CANCLE寄存器寫1；
	5. 讀取當前中斷的配置資訊和中斷回應狀態（IRQ\_STATUS）
3. 設置處理器的中斷回應優先順序閾值（STATUS32寄存器的E[3：0]域）
4. 開啟全域中斷。通過SETI指令實現，將對STATUS32寄存器的IE域置1

以下是一段配置中斷的組合語言程式示例。

|  |
| --- |
| ; ------------------------------------------------------; Interrupt programming constants;.equ irq\_priority\_to\_0, 0x00000000 ; priority-0.equ trigger\_level, 0x00000000 ; level-sensitive.equ irq\_sel\_number, 0x00000012 ; 18.equ EALL, 0x1e ; E[3:0]=4'b1111.equ irq\_en, 0x1 ; interrupt enable; ------------------------------------------------------; programming the interrupt unit; disable global interrupt clri tmp1\_var\_reg ; clearing all interrupts before programming; configuring the interrupt register bank sr irq\_sel\_number, [IRQ\_SELECT] ; Setting Interrupt Number to 18 sr trigger\_level, [IRQ\_TRIGGER] ; level-sensitizing the interupt sr irq\_priority\_to\_0, [IRQ\_PRIORITY]; Setting the Prioirty of irq18 as 0 flag EALL ; enable interrupts at all priority levels sr irq\_en, [IRQ\_ENABLE] ; local enable for the specific irq18 interrupt; enable global interrupt  seti 0 ; Setting the global enable for the interupt unit IE bit ; [5:4] == 0, indicates enable interrupts  ; without changing enable level |

# 附錄二 Baremetal IO介面函數清單

## 一、PIN\_MUX介面函數

1. **mux\_init**

原型：

 ***void mux\_init(DWCREG\_PTR muxRegs)***

功能：

將Pin Mux模組PMOD\_MUX\_CTRL/SPI\_MAP\_CTRL/UART\_MAP\_CTRL寄存器配置為初始值。

參數：

 ***muxRegs*** 輸入，Pin Mux模組基底位址

1. **set\_pmod\_mux**

原型：

 ***void set\_pmod\_mux(DWCREG\_PTR muxRegs, unsigned int val)***

功能：

 配置Pin Mux模組PMOD\_MUX\_CTRL寄存器

參數：

 ***muxRegs*** 輸入，Pin Mux模組基底位址

 ***val***  輸入，配置值

1. **get\_pmod\_mux**

原型：

 ***unsigned int get\_pmod\_mux(DWCREG\_PTR muxRegs)***

功能：

 獲取Pin Mux模組PMOD\_MUX\_CTRL寄存器值

參數：

 ***muxRegs*** 輸入，Pin Mux模組基底位址

返回：

 返回獲取的PMOD\_MUX\_CTRL寄存器值

1. **change\_pmod\_mux**

原型：

 ***void change\_pmod\_mux(DWCREG\_PTR muxRegs, unsigned int val, unsigned int change\_bits)***

功能：

 更改Pin Mux模組PMOD\_MUX\_CTRL寄存器值

參數：

 ***muxRegs*** 輸入，Pin Mux模組基底位址

 ***val***  輸入，配置值

 ***bits***  輸入，配置遮罩

注意：

 配置值見如下運算式：

muxRegs[PMOD\_MUX\_CTRL] = ((muxRegs[PMOD\_MUX\_CTRL] & ~change\_bits) | val)

1. **set\_spi\_mux**

原型：

 ***void set\_spi\_map(DWCREG\_PTR muxRegs, unsigned int val)***

功能：

 配置Pin Mux模組SPI\_MAP\_CTRL寄存器

參數：

 ***muxRegs*** 輸入，Pin Mux模組基底位址

 ***val***  輸入，配置值

1. **get\_spi\_map**

原型：

 ***unsigned int get\_spi\_map(DWCREG\_PTR muxRegs)***

功能：

 獲取Pin Mux模組SPI\_MAP\_CTRL寄存器值

參數：

 ***muxRegs*** 輸入，Pin Mux模組基底位址

返回：

 返回獲取的SPI\_MAP\_CTRL寄存器值

1. **set\_uart\_map**

原型：

 ***void set\_uart\_map(DWCREG\_PTR muxRegs, unsigned int val)***

功能：

 配置Pin Mux模組UART\_MAP\_CTRL寄存器

參數：

 ***muxRegs*** 輸入，Pin Mux模組基底位址

 ***val***  輸入，配置值

1. **get\_uart\_map**

原型：

 ***unsigned int get\_uart\_map(DWCREG\_PTR muxRegs)***

功能：

 獲取Pin Mux模組UART\_MAP\_CTRL寄存器值

參數：

 ***muxRegs*** 輸入，Pin Mux模組基底位址

返回：

 返回獲取的UART\_MAP\_CTRL寄存器值

## 二、I2C介面函數

1. **i2c\_init**

原型：

***void i2c\_init (***

***DWCREG\_PTR i2cRegs,***

***unsigned int speed,***

***unsigned char slave\_addr***

***)***

功能：

初始化I2C控制器並設置從設備位址

參數：

i2c\_Regs 指標變數指向i2c配置寄存器的位址

speed 無符號整型變數指定i2c的速度模式

slave\_addr 無符號字元變數指定從設備位址

返回：

無

1. **i2c\_write**

原型：

***int i2c\_write(***

 ***DWCREG\_PTR i2cRegs,***

 ***char\* buf,***

 ***int length***

***)***

功能：

向I2C從設備發送命令

參數：

i2c\_Regs 指標變數指向i2c配置寄存器的位址

buf 字元型指標變數指向待發送命令的起始位址

length 整型變數指定待發送命令字節長度

返回：

int txcount 此次發送的位元組數目

1. **i2c\_read**

原型：

***int i2c\_read(***

 ***DWCREG\_PTR i2c,***

 ***char\* data,***

 ***int length,***

 ***unsigned int timeout***

***)***

功能：

向從設備發起讀命令並讀取資料

參數：

i2c 指標變數指向i2c配置寄存器的位址

data 字元型指標指向接收讀取資料的緩存起始位址

length 整型變數指定讀取資料的長度

timeout 無符號整型變數指定指定讀取資料時等待的最大時長

返回：

0 讀取資料成功

1 讀取超時，失敗

## 三、GPIO介面函數

1. **gpio\_init**

原型：

***void gpio\_init (***

***DWCREG\_PTR gpioRegs,***

***)***

功能：

初始化按鍵，LED燈和撥碼開關

參數：

gpioRegs 指標變數指向gpio的起始位址

返回：

無

1. **gpio\_get\_buttons**

原型：

***unsigned gpio\_get\_buttons(***

 ***DWCREG\_PTR gpioRegs,***

***)***

功能：

獲取按鍵狀態

參數：

gpioRegs 指標變數指向gpio的起始位址

返回：

~gpioRegs[EXT\_PORTA] 按鍵狀態

1. **gpio\_get\_switches**

原型：

***unsigned gpio\_get\_switches(***

 ***DWCREG\_PTR gpioRegs,***

***)***

功能：

獲取撥碼開關狀態

參數：

gpioRegs 指標變數指向gpio的起始位址

返回：

~gpioRegs[EXT\_PORTC] 撥碼開關狀態

1. **gpio\_set\_leds**

原型：

***unsigned gpio\_set\_leds(***

 ***DWCREG\_PTR gpioRegs,***

 ***unsigned int leds***

***)***

功能：

設置LED燈亮滅狀態

參數：

gpioRegs 指標變數指向gpio的起始位址

leds 無符號整型變數指定LED燈的亮滅狀態

返回：

無

1. **gpio\_get\_switches**

原型：

**unsigned int gpio\_getleds(**

 **DWCREG\_PTR gpioRegs,**

**)**

功能：

獲取LED燈亮滅狀態

參數：

gpioRegs 指標變數指向gpio的起始位址

返回：

int gpioRegs[SWPORTB\_DR] LED燈亮滅狀態

## 四、SPI 介面函數

1. **spi\_init**

原型：

***void uart\_consoleInit (***

 ***DWCFG\_PTR spiRegs,***

 ***unsigned int speed,***

 ***unsigned int polarity***

***);***

功能：

初始化SPI控制器

參數：

spiRegs 指標變數指向spi配置寄存器的位址

speed spi串列傳輸速率

polarity spi串列時鐘極性，高有效or低有效

返回：

無

1. **spi\_clearFifo**

原型：

*void spi\_clearFifo(*

 *DWCREG\_PTR spi*

*);*

功能：

等待SPI接收Fifo被清空

參數：

spi 指標變數指向spi配置寄存器的位址

返回：

無

1. **spi\_waitTr**

原型：

***void spi\_waitTr(***

 ***DWCREG\_PTR spi***

***);***

功能：

等待spi發送資料完畢

參數：

spi 指標變數指向spi配置寄存器的位址

返回：

無

1. **spi\_sendCmd**

原型：

***unsigned int spi\_sendCmd(***

***DWCREG\_PTR spi,***

***unsigned int slaveNum,***

***DATA\_BUF\_PTR pTxBuf,***

***unsigned int numTr,***

***unsigned int offsetRx***

***);***

功能：

向SPI從設備發送資料並讀取返回資料

參數：

spi 指標變數指向spi配置寄存器的位址

slavaNum 從設備設備號

pTxBuf 指標變數指向待發送資料的起始位址

numTr 無符號整型變數指定待發送或接收資料的位元組長度

offsetRx 無符號整型變數指定接收資料偏移量，即有效資料的偏移位元組

返回：

reccount 接收到的有效資料位元組數

## 五、UART介面函數

1. **uart\_initDevice**

原型：

***void uart\_initDevice (***

 ***DWCREG\_PTR uartRegs,***

 ***uart\_baudrate\_t baud,***

 ***uart\_data\_bits\_t data\_bits,***

 ***uart\_stop\_t stop,***

 ***uart\_parity\_t parity***

***);***

功能：

初始化UART控制器

參數：

uartRegs 指標變數指向uart配置寄存器的位址

baud 枚舉變數指定uart傳輸串列傳輸速率

data\_bits 枚舉變數指定uart傳輸有效資料的位元數

stop 枚舉變數指定uart傳輸停止位元 的位數

parity 枚舉變數指定uart傳輸校驗位元類型

返回：

無

1. **uart\_print**

原型：

***void uart\_print(***

 ***DWCREG\_PTR uartRegs,***

 ***const char\* pBuf***

***);***

功能：

使用串口列印一段字串資訊

參數：

uartRegs 指標變數指向uart配置寄存器的位址

pBuf 字元型指標變數指向待列印字串的起始位址

返回：

無

1. **uart\_printHex**

原型：

***void uart\_printHex(***

 ***DWCREG\_PTR uartRegs,***

 ***unsigned int hex***

***);***

功能：

使用串口列印32位元16進制數位

參數：

uartRegs 指標變數指向uart配置寄存器的位址

hex 無符號整型變數指定待列印的16進制數位

返回：

無

1. **uart\_printUInt8**

原型：

***void uart\_printUInt8(***

 ***DWCREG\_PTR uartRegs,***

 ***unsigned char hex***

***);***

功能：

使用串口列印8位元16進制數位

參數：

uartRegs 指標變數指向uart配置寄存器的位址

hex 無符號字元型變數指定待列印的16進制數位

返回：

無

1. **uart\_consoleInit**

原型：

***void uart\_consoleInit ();***

功能：

初始化UART控制器

參數：

無

返回：

無

1. **uart\_consoleMsg**

原型：

***void uart\_consoleMsg(***

 ***const char\* pBuf***

***);***

功能：

使用串口列印一段字串資訊

參數：

pBuf 字元型指標變數指向待列印字串的起始位址

返回：

無

1. **uart\_consolePtrHex**

原型：

***void uart\_consolePtrHex(***

 ***unsigned int hex***

***);***

功能：

使用串口列印32位元16進制數位

參數：

hex 無符號整型變數指定待列印的16進制數位

返回：

無

1. **uart\_consolePrtUint8**

原型：

***void uart\_consolePtrUint8(***

***unsigned char hex***

***);***

功能：

使用串口列印8位元16進制數位

參數：

hex 無符號字元型變數指定待列印的8位元16進制數位

返回：

無

## 六、LCD介面函數

參考資料：PmodCLS\_rm\_RevD-E\_1.pdf

http://www.digilentinc.com/Data/Products/PMOD-CLS/PmodCLS\_rm\_RevD-E\_1.pdf

1. **Lcd\_init**

原型：

***void Lcd\_Init (DWCREG\_PTR spi\_master)***

功能：

 初始化Lcd控制器

參數：

 ***spi\_master*** 輸入，連接Lcd的SPI主方控制寄存器位址

注意：

 程式中使用Lcd前均需要使用該函數將Lcd控制器初始化到可用狀態

1. **Lcd\_ClearScreen**

原型：

***void Lcd\_ClearScreen (void)***

功能：

 清除輸出到Lcd上的內容

參數：

 無

1. **Lcd\_Home**

原型：

***void Lcd\_Home (void)***

功能：

 將輸出游標移動到螢幕Home(0行0列)位置

參數：

 無

注意：

 該函數不會擦出螢幕內容，只會移動游標，用於控制後面的文字輸出或文字擦除

1. **Lcd\_Cursor**

原型：

***void Lcd\_Cursor (uint8\_t row, uint8\_t col)***

功能：

 將輸出游標移動到螢幕Home(0行0列)位置

參數：

 無

1. **Lcd\_CursorOff**

原型：

***void Lcd\_CursorOff (void)***

功能：

 在Lcd螢幕上不顯示游標

參數：

 無

1. **Lcd\_ClearRow**

原型：

***void Lcd\_ClearRow (uint8\_t row)***

功能：

 清除Lcd螢幕指定行的全部內容

參數：

 ***row*** 輸入，特定行數，從上到下以0開始計數

1. **Lcd\_DisplayMode**

原型：

***void Lcd\_DisplayMode (bool onoff)***

功能：

 設置Lcd顯示模式

參數：

 ***onoff*** 輸入，指定模式

 ***0：***每16個字元換行

 ***1：*** 每40個字元換行

1. **Lcd\_DisplayOnOff**

原型：

***void Lcd\_DisplayOnOff (bool onoff)***

功能：

 設置Lcd顯示模式

參數：

 ***Onoff*** 輸入，關閉或打開Lcd顯示

 ***0：***關閉Lcd顯示

 ***1：*** 打開Lcd顯示

1. **Lcd\_DisplayString**

原型：

***void Lcd\_DisplayString (uint8\_t row, uint8\_t col, char \*string)***

功能：

 在特定行和列座標處(***row:col***)開始顯示字串

參數：

 ***row*** 輸入，顯示起始點行座標

 ***col*** 輸入，顯示起始點列座標

 ***string*** 輸入，字串緩衝區

注意：

1. 一次顯示的字元不能超過Lcd列數
2. 顯示過程中遇到字串結束符***’\0’***停止顯示
3. 顯示過程中遇到控制字元***’\r’***或***’\n’***停止顯示