Command Queue User Guide

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Author** | **Description** |
| 0.1 | 2017/08/10 | Kevin.cheng | First Version |
| 0.2 | 2017/08/14 | Kevin.cheng | Remove do\_cmdq\_alignment callback function.  Add Trigger event |
| 0.3 | 2017/08/21 | Kevin.cheng | Add more callback API. |
| 0.4 | 2017/08/23 | Kevin.cheng | Add cmdq\_poll\_reg\_bits callback function for poll\_eq mode and poll\_neq mode. |
| 0.5 | 2017/08/30 | Kevin.cheng | Follow MHAL coding style |
| 0.6 | 2017/09/04 | Kevin.cheng | Modify following API  MHAL\_CMDQ\_IsCmdqEmptyIdle  MHAL\_CMDQ\_ReadStatusCmdq  MHAL\_CMDQ\_ReadDummyRegCmdq  Add API  MHAL\_CMDQ\_ClearTriggerEvent |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Overview**

在I2 platform 有5個command queue，如圖1.1所示，每個CMDQ分配如下，VPE一個、DIVP一個、H264一個、H265兩個。

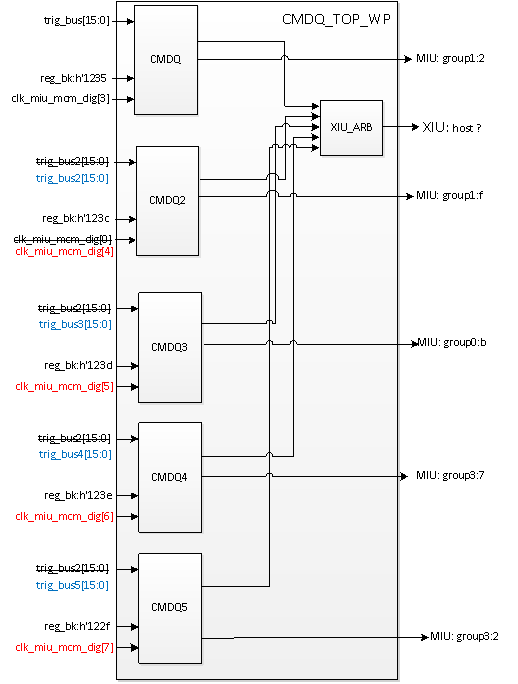


圖1-1

* Trigger bus definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | CMDQ1 | CMDQ2 | CMDQ3 | CMDQ4 | CMDQ5 |
| **ID** | **source** | **source** | **source** | **source** | **source** |
| [15] | S0\_intr\_mdw\_write\_done\_d | dmagen\_cmdq\_trig1 | bdma\_cmdq\_trig1 | dmagen\_cmdq\_trig1 | S0\_intr\_mdw\_write\_done\_d |
| [14] | S0\_intr\_mgw\_fire | dmagen\_cmdq\_trig0 | bdma\_cmdq\_trig0 | dmagen\_cmdq\_trig0 | bdma\_cmdq\_trig0 |
| [13] | Reg\_cmdq\_dummy[15] | S1\_intr\_mdw\_write\_done\_d | ive\_cmdq\_trig | S1\_intr\_mdw\_write\_done\_d | ive\_cmdq\_trig |
| [12] | ldc\_cmdq\_trig | S1\_intr\_mgw\_fire | ldc\_cmdq\_trig | S1\_intr\_mgw\_fire | S0\_intr\_mgw\_fire |
| [11] | ge\_cmdq\_trig | Reg\_cmdq2\_dummy[15] | ge\_cmdq\_trig | Reg\_cmdq4\_dummy[15] | Reg\_cmdq5\_dummy[15] |
| [10] | core1\_mhe\_cmdq\_trig | core1\_mhe\_cmdq\_trig | core1\_mhe\_cmdq\_trig | core1\_mhe\_cmdq\_trig | core1\_mhe\_cmdq\_trig |
| [9] | core0\_mhe\_cmdq\_trig | core0\_mhe\_cmdq\_trig | core0\_mhe\_cmdq\_trig | core0\_mhe\_cmdq\_trig | core0\_mhe\_cmdq\_trig |
| [8] | core1\_mfe\_cmdq\_trig | core1\_mfe\_cmdq\_trig | core1\_mfe\_cmdq\_trig | core1\_mfe\_cmdq\_trig | core1\_mfe\_cmdq\_trig |
| [7] | core0\_mfe\_cmdq\_trig | core0\_mfe\_cmdq\_trig | core0\_mfe\_cmdq\_trig | core0\_mfe\_cmdq\_trig | core0\_mfe\_cmdq\_trig |
| [6] | dip\_cmdq\_trig | dip\_cmdq\_trig | dip\_cmdq\_trig | dip\_cmdq\_trig | dip\_cmdq\_trig |
| [5] | gop\_cmdq\_trig4 | gop\_cmdq\_trig4 | gop\_cmdq\_trig4 | gop\_cmdq\_trig4 | gop\_cmdq\_trig4 |
| [4] | gop\_cmdq\_trig2 | gop\_cmdq\_trig2 | gop\_cmdq\_trig2 | gop\_cmdq\_trig2 | gop\_cmdq\_trig2 |
| [3] | gop\_cmdq\_trig013 | gop\_cmdq\_trig013 | gop\_cmdq\_trig013 | gop\_cmdq\_trig013 | gop\_cmdq\_trig013 |
| [2] | sc\_cmdq\_trig2 | sc\_cmdq\_trig2 | sc\_cmdq\_trig2 | sc\_cmdq\_trig2 | sc\_cmdq\_trig2 |
| [1] | sc\_cmdq\_trig013 | sc\_cmdq\_trig013 | sc\_cmdq\_trig013 | sc\_cmdq\_trig013 | sc\_cmdq\_trig013 |
| [0] | isp\_cmdq\_trig | isp\_cmdq\_trig | isp\_cmdq\_trig | isp\_cmdq\_trig | isp\_cmdq\_trig |
|  |  |  |  |  |  |
| Note |  |  |  |  |  |

**Command Queue Interface**

* CMDQ service public API定義如下

|  |  |
| --- | --- |
| **Purpose** | To get cmdq service interface and allocate cmdq buffer and mload buffer. |
| **Syntax** | MHAL\_CMDQ\_CmdqInterface\_t \*MHAL\_CMDQ\_GetSysCmdqService(MHAL\_CMDQ\_ID eCmdqId, MHAL\_CMDQ\_BufDescript\_t \*pCmdqBufDesp ,MS\_BOOL bForceRIU) |
| **Parameters** | /\*cmdq ID\*/  typedef enum  {  E\_MHAL\_CMDQ\_ID\_VPE = 0,  E\_MHAL\_CMDQ\_ID\_DIVP,  E\_MHAL\_CMDQ\_ID\_H265\_VENC0,  E\_MHAL\_CMDQ\_ID\_H265\_VENC1,  E\_MHAL\_CMDQ\_ID\_H264\_VENC0,  E\_MHAL\_CMDQ\_ID\_MAX  }MHAL\_CMDQ\_ID;  /\*allocate cmdq buffer and mload buffer\*/  typedef struct MHAL\_CMDQ\_BufDescript\_s  {  MS\_U32 u32CmdqBufSize;  MS\_U16 u32CmdqBufSizeAlign;  MS\_U32 u32MloadBufSize;  MS\_U16 u16MloadBufSizeAlign;  }MHAL\_CMDQ\_BufDescript\_t;  /\*True is RIU mode , False is CMDQ mode\*/  MS\_BOOL bForceRIU |
| **Return** | Return MHAL\_CMDQ\_CmdqInterface\_t point, NULL is fail. |

|  |  |
| --- | --- |
| **Purpose** | Release cmdq service |
| **Syntax** | void MHAL\_CMDQ\_ReleaseSysCmdqService(MHAL\_CMDQ\_ID eCmdqId) |
| **Parameters** | /\*cmdq ID\*/  typedef enum  {  E\_MHAL\_CMDQ\_ID\_VPE = 0,  E\_MHAL\_CMDQ\_ID\_DIVP,  E\_MHAL\_CMDQ\_ID\_H265\_VENC0,  E\_MHAL\_CMDQ\_ID\_H265\_VENC1,  E\_MHAL\_CMDQ\_ID\_H264\_VENC0,  E\_MHAL\_CMDQ\_ID\_MAX  }MHAL\_CMDQ\_ID; |
| **Return** | void |

* CMDQ Interface API定義如下

struct MHAL\_CMDQ\_CmdqInterface\_s

{

//menuload ring buffer dynamic allocation service

MS\_S32 (\*MHAL\_CMDQ\_GetNextMlodRignBufWritePtr)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf,MS\_PHYADDR\* phyWritePtr);

MS\_S32 (\*MHAL\_CMDQ\_UpdateMloadRingBufReadPtr)(MHAL\_CMDQ\_CmdqInterface\_t\* pCmdinf, MS\_PHYADDR phyReadPtr);

MS\_S32 (\*MHAL\_CMDQ\_MloadCopyBuf)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf, void \* MloadBuf, MS\_U32 u32Size, MS\_U16 u16Alignment, MS\_PHYADDR \*phyRetAddr);

MS\_S32 (\*MHAL\_CMDQ\_CheckBufAvailable)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf , MS\_U32 u32CmdqNum);

MS\_S32 (\*MHAL\_CMDQ\_WriteDummyRegCmdq)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf, MS\_U16 u16Value);

MS\_S32 (\*MHAL\_CMDQ\_ReadDummyRegCmdq)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf,MS\_U16\* u16RegVal);

MS\_S32 (\*MHAL\_CMDQ\_WriteRegCmdqMask)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf, MS\_U32 u32RegAddr, MS\_U16 u16Value,MS\_U16 u16WriteMask);

MS\_S32 (\*MHAL\_CMDQ\_WriteRegCmdq)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf, MS\_U32 u32RegAddr, MS\_U16 u16Value);

MS\_S32 (\*MHAL\_CMDQ\_CmdqPollRegBits)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf, MS\_U32 u32RegAddr, MS\_U16 u16Value, MS\_U16 u16WriteMask,MS\_BOOL bPollEq);

MS\_S32 (\*MHAL\_CMDQ\_CmdqAddWaitEventCmd)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf, MHAL\_CMDQ\_EVENT\_ID eEvent);

MS\_S32 (\*MHAL\_CMDQ\_CmdqAbortBuffer)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf);

MS\_S32 (\*MHAL\_CMDQ\_CmdqResetEngine)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf);

MS\_S32 (\*MHAL\_CMDQ\_ReadStatusCmdq)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf,MS\_U32\* u32StatVal); MS\_S32 (\*MHAL\_CMDQ\_KickOffCmdq)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf);

MS\_S32 (\*MHAL\_CMDQ\_IsCmdqEmptyIdle)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf,MS\_BOOL\* bIdleVal)

MS\_S32(\*MHAL\_CMDQ\_ClearTriggerEvent)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf)

VOID \* pCtx;

};

|  |  |
| --- | --- |
| **Purpose** | In non-RIU mode : 得到目前menuload buffer的目前write address.  In RIU mode , it is available. |
| **Syntax** | MS\_S32 (\*MHAL\_CMDQ\_GetNextMlodRignBufWritePtr)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf,MS\_PHYADDR\* phyWritePtr) |
| **Parameters** | MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf  /\*Output phyWritePoint\*/  MS\_PHYADDR\* phyWritePtr |
| **Return** | 0 is success , other is fail |

|  |  |
| --- | --- |
| **Purpose** | In non-RIU mode : 更新目前menuload buffer的目前read address.  In RIU mode , it is available. |
| **Syntax** | MS\_S32 (\*MHAL\_CMDQ\_UpdateMloadRingBufReadPtr)(MHAL\_CMDQ\_CmdqInterface\_t\* pCmdinf, MS\_PHYADDR phyReadPtr) |
| **Parameters** | MHAL\_CMDQ\_CmdqInterface\_t\* pCmdinf //cmdq interface  MS\_PHYADDR phyReadPtr //update 的read point |
| **Return** | 0 is success , other is fail |

|  |  |
| --- | --- |
| **Purpose** | In non-RIU mode : 將需要menuload的內容，copy至cmdq service的mload buffer.  In RIU mode : it is available. |
| **Syntax** | MS\_S32 (\*MHAL\_CMDQ\_MloadCopyBuf)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf, void \* MloadBuf, MS\_U32 u32Size, MS\_U16 u16Alignment, MS\_PHYADDR \*phyRetAddr) |
| **Parameters** | HAL\_CMDQ\_CmdqInterface\_t \*pCmdinf//mload interface  void \* MloadBuf //要copy到 cmdq service的mload buffer 的source point.  MS\_U32 u32Size // source buffer的 size  u16Alignment //alignment size  MS\_PHYADDR \*phyRetAddr //return phyAddr |
| **Return** | 0 is success , other is fail |

|  |  |
| --- | --- |
| **Purpose** | In non-RIU mode :每次要進行write COMQ動作時，需呼叫此function來確保cmdq buffer是足夠的。  In RIU mode : it is unavailable. |
| **Syntax** | MS\_S32 (\*MHAL\_CMDQ\_CheckBufAvailable)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf , MS\_U32 u32CmdqNum) |
| **Parameters** | /\*cmdq interface point\*/  MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf  /\*需要的cmdq 數量\*/  MS\_U32 u32CmdqNum |
| **Return** | if is success , will return current cmdq available cmdq number.  0 is fail. |

|  |  |
| --- | --- |
| **Purpose** | In non-RIU mode : 將write command 的register address and value寫進cmdq.  In RIU mode : it will write register directly. |
| **Syntax** | MS\_S32 (\*MHAL\_CMDQ\_WriteRegCmdqMask)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf, MS\_U32 u32RegAddr, MS\_U16 u16Value,MS\_U16 u16WriteMask) |
| **Parameters** | /\*cmdq interface point\*/  MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf  /\*register address\*/  MS\_U32 u32RegAddr  /\*register value\*/  MS\_U16 u16Value  /\*register value mask\*/  MS\_U16 u16WriteMask  For example :  value = 1111\_1111\_1111\_1111  write\_mask = 1111\_1111\_0000\_0000  real write value = 1111\_1111\_0000\_0000 |
| **Return** | 0 is success , other is fail. |

|  |  |
| --- | --- |
| **Purpose** | In non-RIU mode : 將write command 的register address and value寫進cmdq.  In RIU mode : it will write register directly. |
| **Syntax** | MS\_S32 (\*MHAL\_CMDQ\_WriteRegCmdq)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf, MS\_U32 u32RegAddr, MS\_U16 u16Value) |
| **Parameters** | /\*cmdq interface point\*/  MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf  /\*register address\*/  MS\_U32 u32RegAddr  /\*register value\*/  MS\_U16 u16Value |
| **Return** | 0 is success , other is fail. |

|  |  |
| --- | --- |
| **Purpose** | In non-RIU mode :將poll command 的register address and polling bits 寫進cmdq.  In RIU mode : it is unavailable. |
| **Syntax** | MS\_S32 (\*MHAL\_CMDQ\_CmdqPollRegBits)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf, MS\_U32 u32RegAddr, MS\_U16 u16Value, MS\_U16 u16WriteMask,MS\_BOOL bPollEq) |
| **Parameters** | /\*cmdq interface point\*/  MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf  /\*register address\*/  MS\_U32 u32RegAddr  /\*register value\*/  MS\_U16 u16Value  /\*register value mask\*/  MS\_U16 u16WriteMask  /\*poll eq or non-eq mode\*/  MS\_BOOL bPollEq  TURE is Poll eq mode  FALSE is poll non-eq mode  For example :  value = 1111\_1111\_1111\_1111  write\_mask = 1111\_1111\_0000\_0000  polling bits = 1111\_1111\_0000\_0000 |
| **Return** | 0 is success , other is fail. |

|  |  |
| --- | --- |
| **Purpose** | In non-RIU mode : 將wait command 的event 寫進cmdq.  In RIU mode : it will add wait command into cmdq buffer and kick off cmdq.Then go to block mode to wait event done. |
| **Syntax** | MS\_S32 (\*MHAL\_CMDQ\_CmdqAddWaitEventCmd)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf, MHAL\_CMDQ\_EVENT\_ID eEvent) |
| **Parameters** | /\*cmdq interface point\*/  MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf  /\*trigger bus event\*/  typedef enum  {  E\_MHAL\_CMDQEVE\_S0\_MDW\_W\_DONE, //Only for cmdq1&cmdq5  E\_MHAL\_CMDQEVE\_S0\_MGW\_FIRE, //Only for cmdq1&cmdq5  E\_MHAL\_CMDQEVE\_S1\_MDW\_W\_DONE, //Only for cmdq2&cmdq4  E\_MHAL\_CMDQEVE\_S1\_MGW\_FIRE, //Only for cmdq2&cmdq4  E\_MHAL\_CMDQEVE\_DMAGEN\_TRIGGER0, //Only for cmdq2&cmdq4  E\_MHAL\_CMDQEVE\_DMAGEN\_TRIGGER1, //Only for cmdq2&cmdq4  E\_MHAL\_CMDQEVE\_BDMA\_TRIGGER0, //Only for cmdq3&cmdq5  E\_MHAL\_CMDQEVE\_BDMA\_TRIGGER1, //Only for cmdq3  E\_MHAL\_CMDQEVE\_IVE\_CMDQ\_TRIG, //Only for cmdq3&cmdq5  E\_MHAL\_CMDQEVE\_LDC\_CMDQ\_TRIG, //Only for cmdq1&cmdq3  E\_MHAL\_CMDQEVE\_GE\_CMDQ\_TRIG, //Only for cmdq1&cmdq3  E\_MHAL\_CMDQEVE\_REG\_DUMMY\_TRIG, //Only for cmdq1&cmdq2&cmdq4&cmdq5  E\_MHAL\_CMDQEVE\_CORE1\_MHE\_TRIG, //Only for ALL  E\_MHAL\_CMDQEVE\_CORE0\_MHE\_TRIG, //Only for ALL  E\_MHAL\_CMDQEVE\_CORE1\_MFE\_TRIG, //Only for ALL  E\_MHAL\_CMDQEVE\_CORE0\_MFE\_TRIG, //Only for ALL  E\_MHAL\_CMDQEVE\_DIP\_TRIG, //Only for ALL  E\_MHAL\_CMDQEVE\_GOP\_TRIG4, //Only for ALL  E\_MHAL\_CMDQEVE\_GOP\_TRIG2, //Only for ALL  E\_MHAL\_CMDQEVE\_GOP\_TRIG013, //Only for ALL  E\_MHAL\_CMDQEVE\_SC\_TRIG2, //Only for ALL  E\_MHAL\_CMDQEVE\_SC\_TRIG013, //Only for ALL  E\_MHAL\_CMDQEVE\_ISP\_TRIG, //Only for ALL  E\_MHAL\_CMDQEVE\_MAX  }MHAL\_CMDQ\_EVENT\_ID; |
| **Return** | 0 is success , other is fail. |

|  |  |
| --- | --- |
| **Purpose** | In non-RIU mode : kick off command queue.  In RIU mode : it is unavailable. |
| **Syntax** | MS\_S32 (\*MHAL\_CMDQ\_KickOffCmdq)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf) |
| **Parameters** | /\*cmdq interface point\*/  MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf |
| **Return** | if is success , will return current cmdq available number. <0 is fail. |

|  |  |
| --- | --- |
| **Purpose** | In non-RIU mode : To check cmdq if is idle.  In RIU mode : it is available. |
| **Syntax** | MS\_S32 (\*MHAL\_CMDQ\_IsCmdqEmptyIdle)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf ,MS\_BOOL\* bIdleVal) |
| **Parameters** | /\*cmdq interface point\*/  MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf  /\* MS\_BOOL\* bIdleVal \*/  Return idle status. |
| **Return** | 0 is success , other is fail. |

|  |  |
| --- | --- |
| **Purpose** | In non-RIU mode : Add write command of dummy register to cmdq.  In RIU mode : write cmdq dummy register directly. |
| **Syntax** | MS\_S32 (\*MHAL\_CMDQ\_WriteDummyRegCmdq)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf, MS\_U16 u16Value) |
| **Parameters** | /\*cmdq interface point\*/  MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf  /\*write value\*/  MS\_U16 u16Value |
| **Return** | 0 is success , other is fail. |

|  |  |
| --- | --- |
| **Purpose** | In non-RIU mode : Read cmdq dummy register.  In RIU mode : it is the same. |
| **Syntax** | MS\_S32 (\*MHAL\_CMDQ\_ReadDummyRegCmdq)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf,MS\_U16\* u16RegVal) |
| **Parameters** | /\*cmdq interface point\*/  MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf  /\*return value\*/  MS\_U16\* u16RegVal |
| **Return** | 0 is success , other is fail. |

|  |  |
| --- | --- |
| **Purpose** | In non-RIU mode : Abort current cmdq write data ,then go back previous .  In RIU mode : it is unavailable. |
| **Syntax** | MS\_S32 (\*MHAL\_CMDQ\_CmdqAbortBuffer)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf) |
| **Parameters** | /\*cmdq interface point\*/  MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf |
| **Return** | 0 is success , other is fail. |

|  |  |
| --- | --- |
| **Purpose** | In non-RIU mode : Read command queue status.  In RIU mode : it is the same. |
| **Syntax** | MS\_S32 (\*MHAL\_CMDQ\_ReadStatusCmdq)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf,MS\_U32\* u32StatVal) |
| **Parameters** | /\*cmdq interface point\*/  MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf  /\* MS\_U32\* u32StatVal \*/  Return Value |
| **Return** | 0 is success , other is fail. |

|  |  |
| --- | --- |
| **Purpose** | In non-RIU mode : Reset cmdq engine and ring buffer.  In RIU mode : it is the same. |
| **Syntax** | MS\_S32 (\*MHAL\_CMDQ\_CmdqResetEngine)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf) |
| **Parameters** | /\*cmdq interface point\*/  MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf |
| **Return** | 0 is success , other is fail. |

|  |  |
| --- | --- |
| **Purpose** | In non-RIU mode : Add write command for clear trigger event  In RIU mode : N/A. |
| **Syntax** | MS\_S32(\*MHAL\_CMDQ\_ClearTriggerEvent)(MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf) |
| **Parameters** | /\*cmdq interface point\*/  MHAL\_CMDQ\_CmdqInterface\_t \*pCmdinf |
| **Return** | 0 is success , other is fail. |

MHAL\_CMDQ\_ClearTriggerEvent() function的使用時機，因為在fire一張frame時user可能去使用wait command 去等待某一個 event，當此event發生時會通知cmdq，此時cmdq應該要將event清除以避免造成下一張frame 等待該event時產生誤動作。

* MHAL\_CMDQ\_ClearTriggerEvent()的Pseudo code



**Command Queue Process**

* How to control command queue service

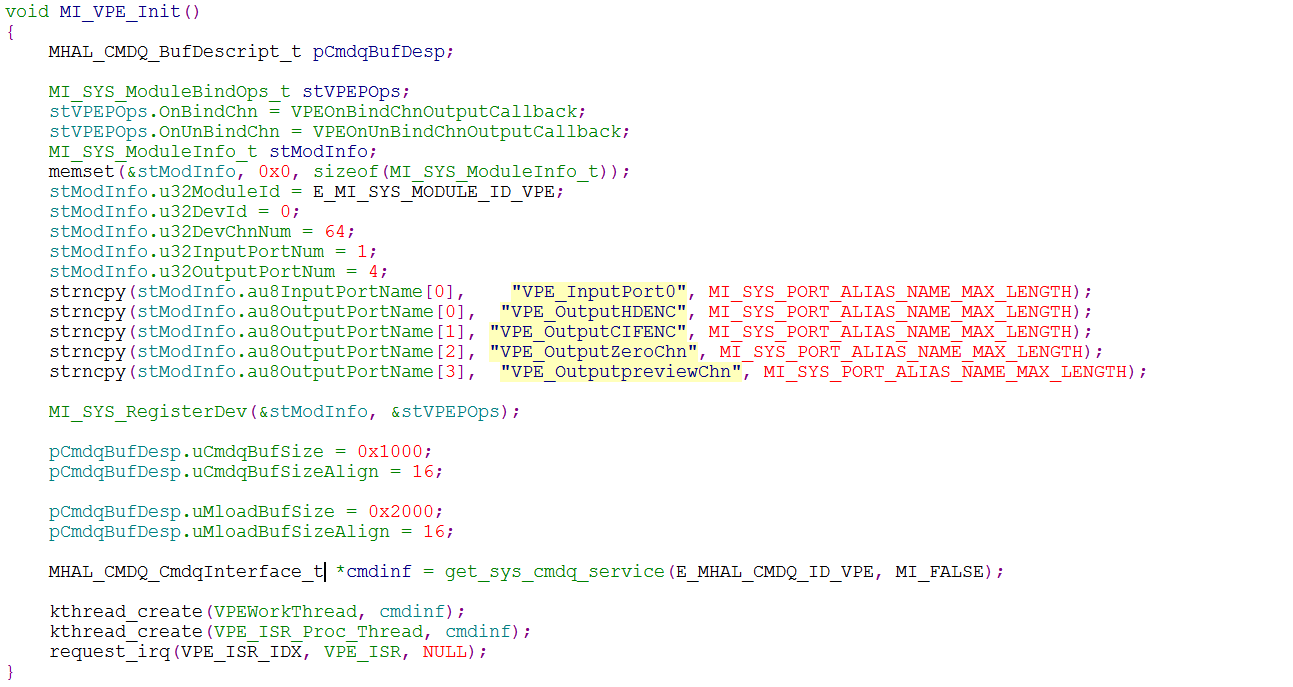
1. 使用者首先需呼叫MHAL\_CMDQ\_GetSysCmdqService 去allocate cmdq buffer及menuload buffer , cmdq service會回傳 cmdq interface structure.
2. 每當要將command寫進command queue buffer之前，都需呼叫MHAL\_CMDQ\_CheckBufAvailable的callback function 來確定command queue buffer是否有足夠的空間來填寫command.
3. 如果有足夠的buffer空間，使用者會將cmdq interface structure所提供 MHAL\_CMDQ\_WriteRegCmdqMask、MHAL\_CMDQ\_WriteRegCmdq、MHAL\_CMDQ\_CmdqPollRegBits、MHAL\_CMDQ\_CmdqAddWaitEventCmd 的callback function 交由負責填寫command的模組來負責。
4. 確定完畢後，由使用者呼叫MHAL\_CMDQ\_KickOffCmdq的callback function 來進行command queue的kick off。

* Menuload callback function的使用

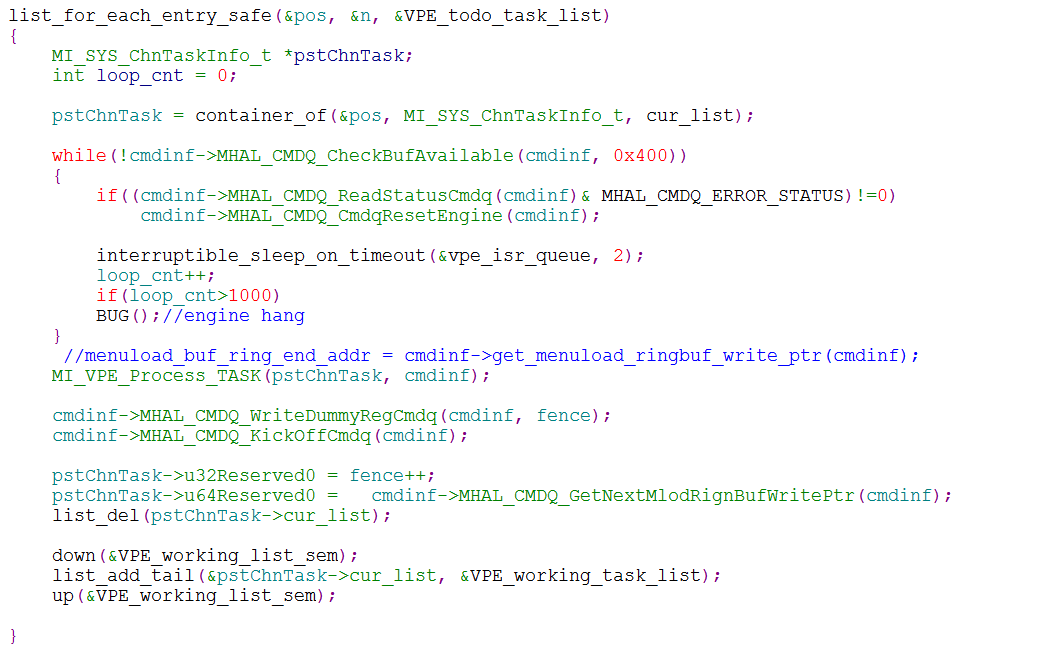
Command queue service只提供menuload的 buffer 管理，主要流程如下

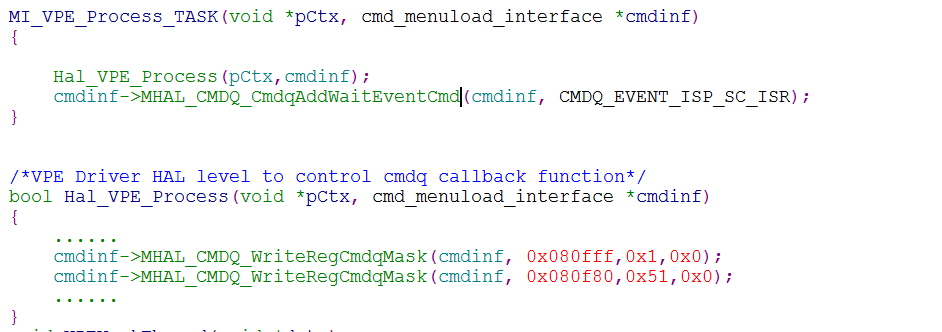
1. 當使用者將cmdq interface structure中MHAL\_CMDQ\_MloadCopyBuf 的callback function交由負責填寫command的模組時，如要使用menuload功能，會先將內容先copy在command queue service的menuload buffer。
2. 如果copy成功，填寫command的模組處理後續驅動menuload IP的動作。
3. 處理完成後，使用者呼叫MHAL\_CMDQ\_GetNextMlodRignBufWritePtr 的callback function來取得menuload buffer write point 。
4. 當command queue執行完畢後，由使用者來呼叫MHAL\_CMDQ\_UpdateMloadRingBufReadPtr來更新menuload buffer的read point，以更新menuload buffer的空間。

* Pseudo code
  + To get command queue service interface



* Process command queue





* ISR Thread to deal with command queue

