SoundWire Quiz

Contributors

葉丞偉　　陳文遠

Content

Q1：如何做 Enumeration?

Master 在進行正常的數據傳輸前，其辨識和初始化 bus 上的 Device 的過程就稱為列舉 (Enumeration)。以下以逐條方式敘述多個裝置列舉的過程：

- 當多個 Slave 剛連上 Bus 時，都會被當作第 0 號設備 (Device Number 0)
- Master 可以發出 Ping Command 來發起裝置列舉
- 所有的 Slave 收到 Ping 後會把自己狀態回應在 Control Word 的 `Per_State_00` (bit 39 & 40) 欄位
    - Slave 應該是要回應 `Attached_OK` (Table 63) 狀態
- 每個 Slave 回應的 `Attached_OK` 會在 bus 上做 wire-OR
- 一旦 Master 看到 `Per_Stat_00` 欄位出現 `Attached_OK`，它就會開始做以下步驟
- 首先，Master 會用 Read Command 讀 Device_ID Register 的第一個 byte (`PCP_DevID_0`)，bus 上所有的 Slave 都會收到該 Read Command
- 所有的 Slave 會在 bus 上回應自己的 `PCP_DevID_0` Value，這些 Value 同樣會在 bus 上做 wire-OR，一旦有 Slave 發現它寫在 bus 上的 Value 與它讀回來看到的不一樣則會退出列舉，直到下次又有 Master 來 Read `PCP_DevID_0`
    - 假如自己是一個 Slave，如果別的 Slave 的 Logic 1 覆蓋掉自己的 Logic 0，此時自己就會檢測到 bus contension 退出此次列舉過程，並且發出 `Command_Ignore`，直到下次又有 Master 來讀 `PCP_DevID_0`
- Device_ID Register 共有 6 bytes (`PCP_DevID_0` to `PCP_DevID_5`)，Master 會依照順序從 `PCP_DevID_0` 讀到 `PCP_DevID_5`，讀到最後應會剩下最後一個 Slave，就賦予該 Slave 新的且唯一的 Device Number (1~11)
    - `PCP_DevID_0` to `PCP_DevID_5` 對應的位址是 `0x0050` to `0x0055`
    - 可以參考 Figure 51，Master 除了賦予 Slave Device Number (1~11) 以外，還可以給一個 `Group_Id` (1~2)。未來 Master 可以利用 Group_Id 同時讀寫多個同群組的 Slave
- 接下來 Master 會重複上面三個步驟直到所有 Slave 都賦予新的 Device Number
- 列舉流程結束

列舉時發生 Bus Clash 屬於正常現象，Slave 不會回 `Command_Failed`。

下圖是每個裝置都有的 DeviceID Register，共 6 bytes (48 bits)，位址從 `0x0050` to `0x0055`：

Register	Device ID	Contents	
SCP_DevID_0 [7:4]	[47:44]	Version	of the implemented SoundWire specification (0b0001 for current spec 1.0)
SCP_DevID_0 [3:0]	[43:40]	Unique ID	4-bit field for “uniquifying” multiple instances of identical parts.
SCP_DevID_1	[39:32]	Manufacturer ID [15:8]	Value assigned by MIPI to members
SCP_DevID_2	[31:24]	Manufacturer ID [7:0]	
SCP_DevID_3	[23:16]	Part ID [15:8]	Assigned by Manufacturer
SCP_DevID_4	[15:8]	Part ID [7:0]	
SCP_DevID_5	[7:0]	Class	0x00

Figure 51 `PCP_DevNumber` Register 位於 Device 的 Address `0x0046`

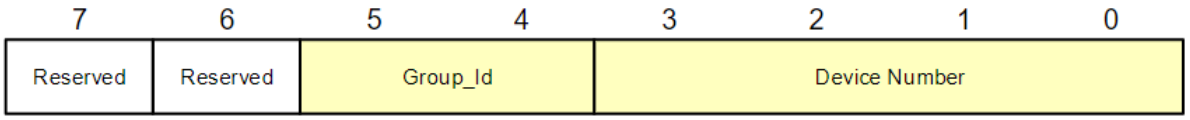


Figure 51 PCP_DevNumber Register

- **`Group_Id`**
  - 1 : Group 1
  - 2 : Group 2
- **`Device Number`**
  - **1 ~ 11 :** Bus 上每個 Slave 唯一的 Device Number
  - **12, 13 :** 12 代表 Group 1；13 代表 Group 2，讓 Master 可以一次 Acceess 多個同群組的 Slave
  - **14 :** 保留給 Master
  - **15 :** 指 Bus 上所有的 Device（Broadcast Access Mode），用於一次性 Write 所有 Device (Broadcast 只對 Write 操作有用)

Q2 : Synchronization 機制

Manager Synchronization

- Manager 會用 Clock Falling Edge 來做為 Frame 的第一個 BitSlot 的起點
- Manager 需要在 Control Word 的 Static Synchronization 欄位生成 `b10110001`
- Manager 需要在 Control Word 的 PHY_Sync 欄位指定要 0 for Basic-PHY 或 1 for High-PHY
- Manager 需要在接下來的 15 個 Frame 中生成 Dynamic Synchronization Code (`1111` to `0111`)
    - 每次 Reset 後 Dynamic Synchronization Code 都要從 `1111` 開始重新生成 (參考 Table 38)

Peripheral Synchronization

Peripheral 從 Unattached 經過一系列 Sync 後到 Attached 的狀態圖:

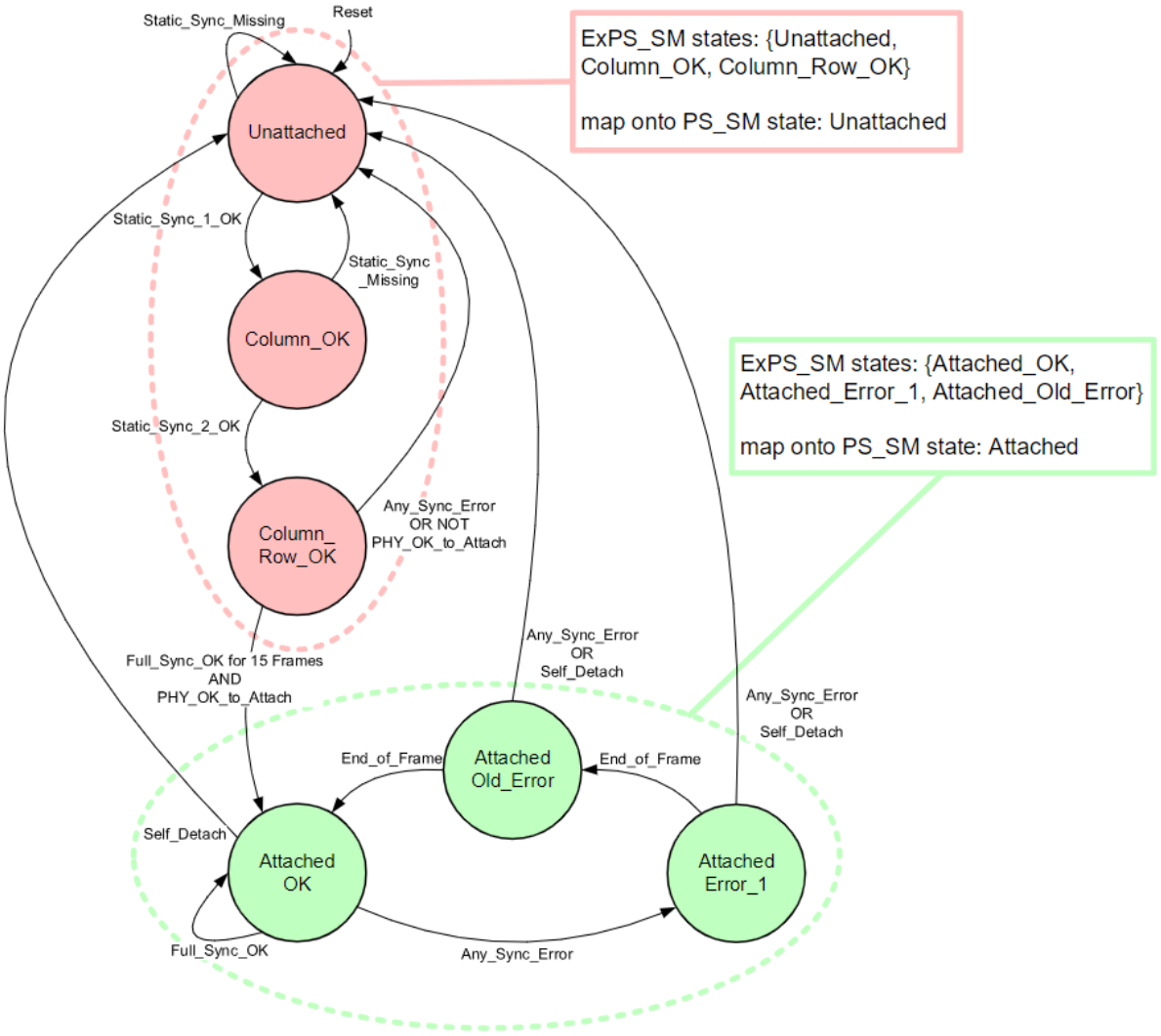


Figure 27 Example Peripheral Synchronization State Machine (ExPS_SM)

1. `Static_Sync_1_OK`: Peripheral 先拿有可能的 NumCol 值去搜尋 Static Sync，若成功搜尋到就可以確認 NumCol 了，並且進到 `Column_OK` State
2. `Static_Sync_Missing`:所有可能的 NumCol 值都搜尋過後仍找不到 Static Sync，狀態轉回到 Unattached
3. `Static_Sync_2_OK`: 接下來 Peripheral 會用已確定 NumCol 值去搜尋 Static Sync，以此來計算出 NumRow(Spec裡的有定義組合)，若也成功的話就會進入 `Column_Row_OK` State
4. `Full_Sync_OK` & `PHY_OK_to_Attach` : Peripheral 用已確定的 NumCol & NumRow 去比對接下來 15 Frames 的 Static/Dynamic Sync 都正確；並且也 Ready 好 Manager 在 `PHY_Sync` 欄位上指定的 PHY (Basic-PHY or /High-PHY)，以上兩者都正確就會進到 `Attached` State
5. `NOT PHY_OK_to_Attach`: Peripheral 在 0x0044 `PCP_Stat` 的 `High-PHY_NotOK` 欄位指示 High-PHY 尚未 Ready 時，狀態會回到 `Unattached`
6. `Any_Sync_Error` : 若 Peripheral 在 Attached 後連續 2 Frames 的 Sync 都錯了 1-bit，就會讓狀態回到 `Unattahced` (一個 Frame 錯 2-bit 以上會直接回 `Unattached` 嗎?)
7. `Self_Detach` : Peripheral 主動 power-down 或因過熱而 shutdown 等原因會讓狀態回到 `Unattahced`

補充："`Unattached`" 狀態到 "`Attached OK`" 狀態需要在 4096 個 Frames 之內就緒。

Q3 : SoundWire Register Mapping

(Hint : 要說一下哪些 address for Data Port / Memory / SDCA control)

Table 75 Top Level Address Map for Registers

Address range	Port Number	Port Name	Prefix for Register names	Notes
0x00000000 – 0x000000FF	0	Control Port & Data Port 0	PCP_ & DP0_	Control and status functions common to the whole Device. Control and status functions specific to Data Port 0
0x00000100 – 0x000001FF	1	Data Port 1	DP1_	Control and status functions specific to Data Port 1
0x00000200 – 0x000002FF	2	Data Port 2	DP2_	Control and status functions specific to Data Port 2
0x00000300 – 0x000003FF	3	Data Port 3	DP3_	Control and status functions specific to Data Port 3
0x00000400 – 0x000004FF	4	Data Port 4	DP4_	Control and status functions specific to Data Port 4
0x00000500 – 0x000005FF	5	Data Port 5	DP5_	Control and status functions specific to Data Port 5
0x00000600 – 0x000006FF	6	Data Port 6	DP6_	Control and status functions specific to Data Port 6

Address range	Port Number	Port Name	Prefix for Register names	Notes
0x00000700 – 0x000007FF	7	Data Port 7	DP7_	Control and status functions specific to Data Port 7
0x00000800 – 0x000008FF	8	Data Port 8	DP8_	Control and status functions specific to Data Port 8
0x00000900 – 0x000009FF	9	Data Port 9	DP9_	Control and status functions specific to Data Port 9
0x00000A00 – 0x00000AFF	10	Data Port 10	DP10_	Control and status functions specific to Data Port 10
0x00000B00 – 0x00000BFF	11	Data Port 11	DP11_	Control and status functions specific to Data Port 11
0x00000C00 – 0x00000CFF	12	Data Port 12	DP12_	Control and status functions specific to Data Port 12
0x00000D00 – 0x00000DFF	13	Data Port 13	DP13_	Control and status functions specific to Data Port 13
0x00000E00 – 0x00000EFF	14	Data Port 14	DP14_	Control and status functions specific to Data Port 14
0x00000F00 – 0x00000FFF	(15)	All Payload Ports	DP_All_P_	Addressing alias used to write or read all of Data Ports 1 through 14 with a single access
0x00001000 – 0x000017FF	—	—	—	(optional) Extended class information 2 Kbytes
0x00001800 – 0x00001FFF	—	—	—	Reserved 2 Kbytes
0x00002000 – 0x0000FFFF	—	—	—	Implementation-defined usage 56 Kbytes
0x00010000 – 0x3FFFFFFF	—	—	—	Implementation-defined usage 1023.96 Mbytes (only accessible using paging registers or BRA)
0x40000000 – 0x43FFFFFF	—	—	—	SoundWire Device Class for Audio Controls 64 Mbytes (see Section 10.1.7) (only accessible using paging registers or BRA)
0x44000000 – 0x47FFFFFF	—	—	—	SoundWire Device Class for Audio memories 64 Mbytes (see Section 10.1.8) (only accessible using paging registers or BRA)
0x48000000 – 0x7FFFFFFF	—	—	—	Reserved 896 Mbytes (only accessible using paging registers or BRA)
0x80000000 – 0xFFFFFFFF	—	—	—	Reserved 2 Gbytes (only accessible using BRA)

Table 76 Address Breakdown within each Port

Offset from Port base address	Registers	Notes (informative)
+0x00 – +0x1F	Non-banked Data Port Registers	Defined by this Specification
+0x20 – +0x2F	Bank 0 Data Port Registers	Defined by this Specification
+0x30 – +0x3F	Bank 1 Data Port Registers	Defined by this Specification
+0x40 – +0x5F	Non-banked Control Port Registers (only in Port 0)	Defined by this Specification
+0x60 – +0x6F	Bank 0 Control Port Registers (only in Port 0)	Defined by this Specification
+0x70 – +0x7F	Bank 1 Control Port Registers (only in Port 0)	Defined by this Specification
+0x40 – +0x7F	Reserved (in Ports 1 – 15)	—
+0x80 – +0x8F	Non-banked Control Port Registers (only in Port 0)	Defined by this Specification
+0x80 – +0x8F	Reserved (in Ports 1 – 15)	—
+0x90 – +0xBF	Reserved	—
+0xC0 – +0xFF	Implementation-defined Registers	Implementation-defined usage

For SDCA Control

0x40000000 – 0x43FFFFFF	SoundWire Device Class for Audio Controls	Defined in ***[MIPI03]***.	Reserved for SoundWire Device Class for Audio Controls *(see 10.2.1 Permission 7)*	64 Mbytes assigned to SDCA Controls. See ***[MIPI03]***.
0x44000000 – 0x47FFFFFF	SoundWire Device Class for Audio Memory	Implementation-defined (see ***[MIPI03]***).	Reserved for SoundWire Device Class for Audio Memory *(see 10.2.1 Permission 8)*	64 Mbytes assigned to memory blocks related to SDCA. See ***[MIPI03]***.
0x48000000 – 0x7FFFFFFF	Reserved	—	Reserved	Reserved 896 Mbytes
0x80000000 – 0xFFFFFFFF	Reserved	—	Reserved	Reserved 2 Gbytes

Hidden register (Implement-defined)

1. RTK Codec的hidden 皆使用 2000h to FFFFh implementation-define MEM Range

0x00002000 – 0x0000FFFF	—	—	—	Implementation-defined usage 56 Kbytes

Q4：如何使用 Paging Register

Case 1: 有Implement PCP_AddrPage1&2 Register

1. 使用16A8D Write Command寫入0x0048&0x0049
2. 接著使用16A8D Access，可以組合出31Bits Address進行操作

   *注意16A bit15 不會被計算到Address 內，取而代之的是PCP_AddrPage1/2 (mapping to actual Addr[30:15])

Case 2: 沒有Implement PCP_AddrPage1&2 Register

1. 僅能使用16A8D Command Access 15bit address range

Address	Name	7	6	5	4	3	2	1	0
0x0048	*PCP_AddrPage1* 32	Addr 30	Addr 29	Addr 28	Addr 27	Addr 26	Addr 25	Addr 24	Addr 23
0x0049	*PCP_AddrPage2* 32	Addr 22	Addr 21	Addr 20	Addr 19	Addr 18	Addr 17	Addr 16	Addr 15

Requirements

1. {3401} A Peripheral shall construct the 32-bit address (Addr[31:0]) used for the Register access resulting from a Read or Write Command according to Table 78.
   Note: the address used for Register accesses resulting from the Bulk Register Access Protocol are specified in 13.2.3 Requirements 11 and 14

Table 78 Address Pages for Read and Write Commands

RegAddr[15] field in Command	Are optional PCP_AddrPage1 & 2 Registers implemented?	Source of Addr[31]	Source of Addr[30:23]	Source of Addr[22:15]	Source of Addr[14:0]
0	—	0	0x00	0x00	RegAddr[14:0]
1	No	0	0x00	0x01	RegAddr[14:0]
1	Yes	0	PCP_AddrPage1	PCP_AddrPage2	RegAddr[14:0]

Q5：TDZ/TZD 是什麼?

t_DZ_Data_Max 及 t_ZD_Data_Min Timing Define 參考下圖:
(DZ: Drive to Hi-Z, ZD: Hi-Z to Drive)

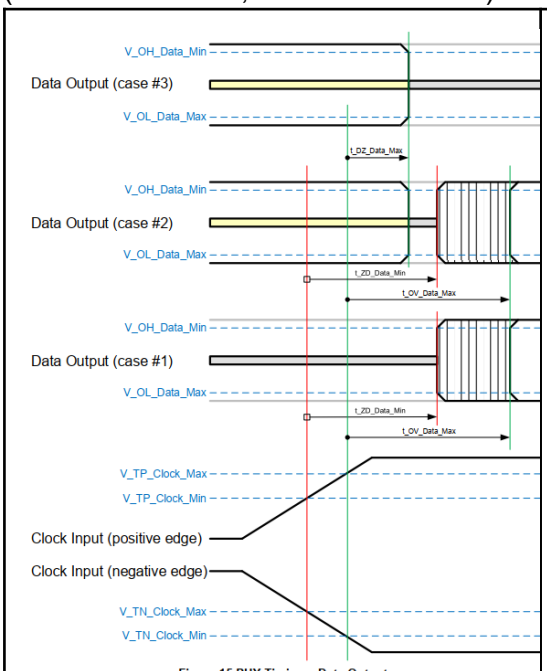


Figure 15 PHY Timing – Data Output

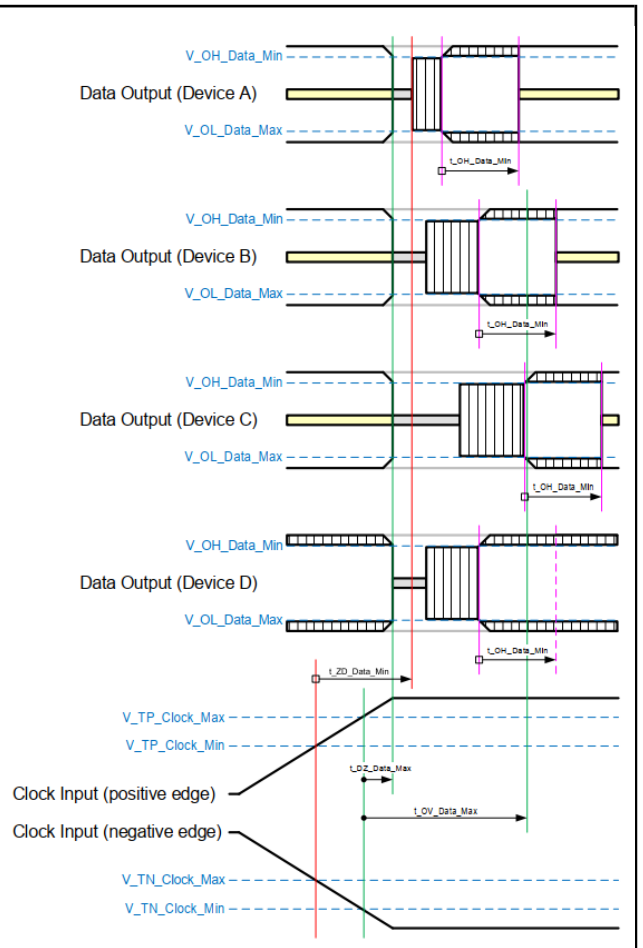


Figure 16 PHY Timing – Data Output with Self-Timed Turn-Off

對應的AC spec

Name	Min.	Typ.	Max.	Units	Notes (informative)
t_DZ_Data_1V8	—	—	4.0	ns	—
t_ZD_Data_1V8	7.9	—	—	ns	—

Note:

1. Bitslut之間設計的Hi-Z空檔區間是為了Multi-Device BUS wire-OR設計
2. Bitslut之間設計的Hi-Z空檔區間，間接限制了SdW頻寬。
   (因為Trace transmission delay存在，Device to Device距離愈長則Turn-around time愈長)

Q6 : Command Response 的 Command_Failed/OK?

NAK and ACK

Control Word 中有 NAK 和 ACK 用來描述對 Command 的 Response。

當有一個設備被選中是 command 的目標時，其 response 的 NAK & ACK 組合可能如下：

Response	NAK	ACK	Remark
`Command_Failed`	1	0	例如奇偶校驗錯誤，或發生 Command Bus Clash Error
`Command_OK`	0	1	例如順利完成 Read/Write 操作
`Command_Ignored`	0	0	例如讀了一個 Write-only Register

當有一個設備沒有被選中是 command 的目標時，其 response 的 NAK & ACK 組合可能如下：

Response	NAK	ACK	Remark
`Command_Failed`	1	0	例如奇偶校驗錯誤，或發生 Command Bus Clash Error
`Command_Ignored`	0	0	例如自己並不是 Read/Write 操作的目標設備

但因為多個 Devices 回應的 Response 會在 data lane 上 wire-OR，因此最後可以歸類出以下四種組合：

Response	NAK	ACK	Remark
`Command_Aborted`	1	0	例如奇偶校驗錯誤，或發生 Command Bus Clash Error
`Command_Aborted`	1	1	例如奇偶校驗錯誤，或發生 Command Bus Clash Error
`Command_OK`	0	1	例如順利完成 Read/Write 操作

Command_Ignored	0	0	例如自己並不是 Read/Write 操作的目標設備，或讀寫到 read/write-only 的暫存器

Command_Aborted

當看到 Command_Aborted Response 時：

- 收到 Write Command 的 Device 不會去做 Write Operation
- 使用 Read Command 的 Manager 會丟掉此次讀到的結果，再重新 Read 一次
- Device 會忽略 Ping Command 的 SSP 欄位
- Manager 會忽略 Ping Command 中的 Slave Status 欄位

Q7：ClockStop Mechanism and ClockStop Prepare

Clock Stopping Mechanism

FSM

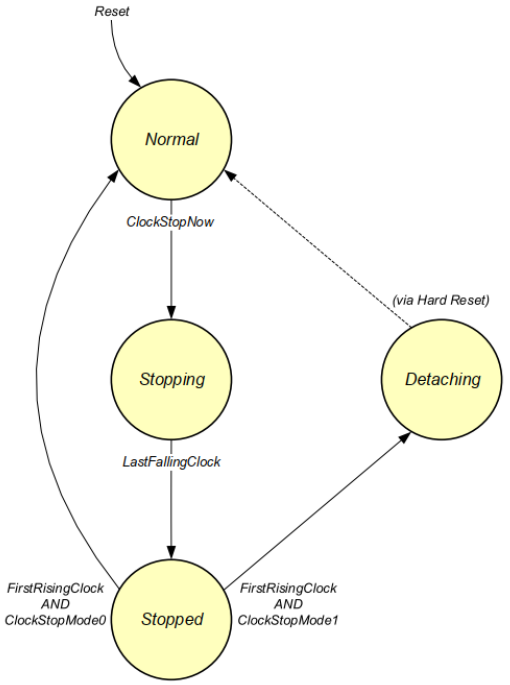


Figure 36 Peripheral Clock Stop State Machine (PCS_SM)

Clock Stop Timing Waveform

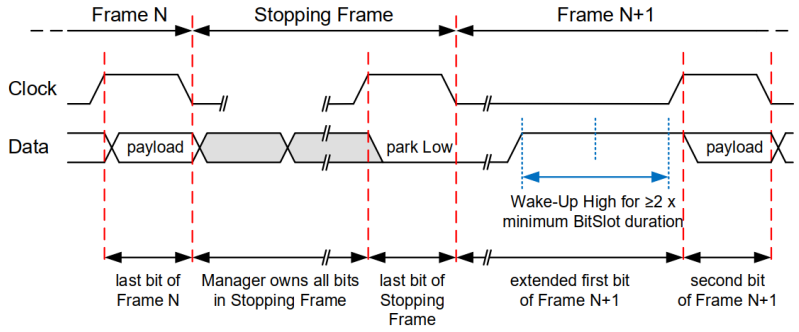


Figure 32 Clock Stop Timing Waveform

整理以下幾點

1. 如何進入 Clock Stop Mode
    a. Manager藉由Clock Stop Prepare 機制告知Peripherial要進入Clock Stop
    b. Manager等待Peripheral一陣子
    c. Manager讀取Peripheral register確認已經就緒，可以進入Clock Stop
    d. Manager明確告知停止的時機
        i. 設定ClockStopNow=1，此時為上圖Frame N
    e. Manager發出 One-Frame wind-down period (Stopping Frame，為上圖Frame N+1)，最後使得Clock & Data 都會處在Low signal level
2. 當處在Clock Stop Mode，Manager 與 Peripheral各自要做甚麼
    a. Manager

i. Manager要持續聆聽Data line是否有任何Wakeup signal (Data line active High 2 BitSluts)

ii. 當聆聽到Wakeup signal後，要等待一陣子
(Clock stopping之後一直到Manager restart clock可能需要額外等待一些padding time, 讓下一個Frame可以對齊)後拉Rising Clock
*注意！**Spec**說明**Manager**喚醒時間沒有要求要多快，**Implement defined**，但是最快是**2 Frame**

b. Peripheral

i. 等待Manager restart Clock

ii. (Optional) If WakeUpEnable bit = 1, 則可以於特定時機自行發起Wakeup

8. {3708} The PCP_SystemCtrl Register shall contain the fields shown in Figure 50.

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	High-PHY_ Select	WakeUp Enable	ClockStop Mode	Reserved	ClockStop Prepare

Figure 50 PCP_SystemCtrl Register

iii. If 處在ClockStopMode0，喚醒後能直接進入Operation State (可以正常收發及串流)

iv. If 處在ClockStopMode1，喚醒後會回到Unattached state，需要重新跑FSM直到重新列舉，並重新由Manager設定payload等設定
(以上兩個ClockStopMode由下方Register控制)

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	High-PHY_ Select	WakeUp Enable	ClockStop Mode	Reserved	ClockStop Prepare

Figure 50 PCP_SystemCtrl Register

Stimulus Name	Description	Notes (informative)
FirstRisingClock	The Peripheral received the rising Clock edge that ended the first BitSlot (PREQ/WakeUp) in the restart Frame.	Peripherals that are configured in ClockStopMode1 will become Unattached.
ClockStopMode0	The ClockStopMode bit in the PCP_SystemCtrl Register has the value 0, or is not implemented *(see 10.2.1 Requirement 15)*	—
ClockStopMode1	The ClockStopMode bit in the PCP_SystemCtrl Register has the value 1	Support for ClockStopMode1 is optional. *(see 8.2.3 Permission 5)*

ClockStop Prepare

FSM參考:

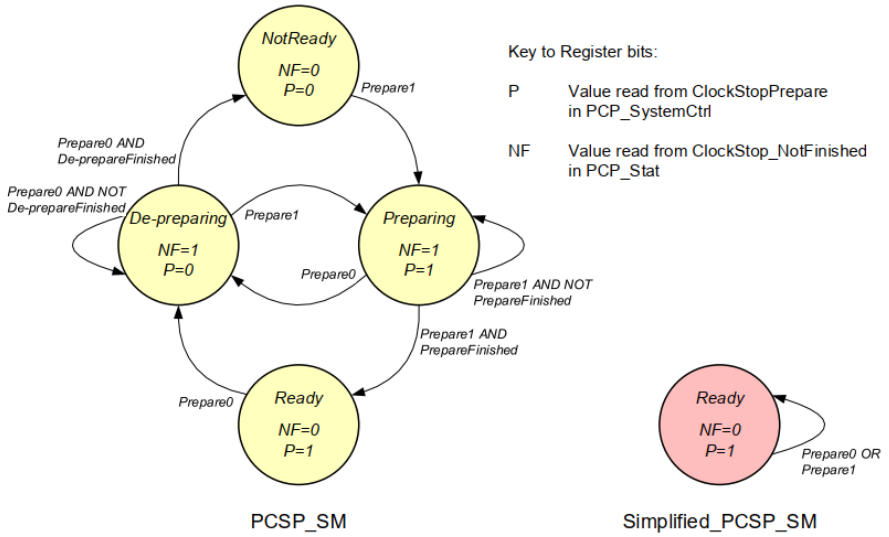


Figure 35 Peripheral Clock Stop Prepare State Machine (PCSP_SM)

Table 48 Stimulus to the Peripheral Clock Stop Prepare State Machine (PCSP_SM)

Stimulus Name	Description	Notes (informative)
Prepare1	The most recent value written to the ClockStopPrepare bit in the PCP_SystemCtrl Register is 1	Read-write register bit in PCSP_SM. Read-only / write-ignored bit for Simplified_PCSP_SM.
Prepare0	The most recent value written to the ClockStopPrepare bit in the PCP_SystemCtrl Register is 0	Read-write register bit in PCSP_SM. Read-only / write-ignored bit for Simplified_PCSP_SM.
Prepare_Finished	The internal operations associated with preparing for the Clock to be stopped have completed	e.g., auxiliary clock source has started and is ready for use. The internal operations are implementation-defined. There might be no operations so that the Prepare completes instantly.
De-prepare_Finished	The internal operations associated with returning to state PCSP_NotReady have completed	e.g., auxiliary clock source has been powered down. The internal operations are implementation-defined. There might be no operations so that the De-prepare completes instantly.

7. {3707} The PCP_Stat Register shall contain the fields shown in Figure 49.

7	6	5	4	3	2	1	0
Reserved	CurrentBank	High-PHY_NotOK	Reserved	Reserved	Reserved	Reserved	ClockStop_NotFinished

Figure 49 PCP_Stat Register

Note: The CurrentBank status bit is described in section 9.1.4. The High-PHY_NotOK bit is specified in 8.2.5. The ClockStop_NotFinished Mode bit is specified in section 8.2.3.

8. {3708} The PCP_SystemCtrl Register shall contain the fields shown in Figure 50.

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	High-PHY_Select	WakeUp Enable	ClockStop Mode	Reserved	ClockStop Prepare

Figure 50 PCP_SystemCtrl Register

Note: The High-PHY_Select bit is specified in 8.2.5. The WakeUpEnable, ClockStopMode, and ClockStopPrepare bits are specified in section 8.2.3.

Step by Step

1. Manager設定Peripheral ClockStopMode 及 WakeUpEnable
2. Manager設定ClockStopPrepare bit=1
3. (Peripheral進行一些Clock的設定讓待會ClockStop後可以持續運作, Ex: enable RC power)
4. Peripheral就緒後, ClockStopNotFinished會清0 (可以用Global alias ID 15來詢問)
5. 此時Manager設定ClockStopNow,

6. {3706} The PCP_Ctrl Register shall contain the fields shown in Figure 48.

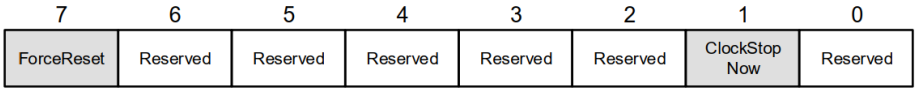


Figure 48 PCP_Ctrl Register

Note: The ForceReset bit is specified in section 8.2.1. The ClockStopNow bit is specified in section 8.2.3.

ClockStop Wakeup

當 Device 處於 ClockStop 時，可以用兩個 bitslot 的 PREQ 來將裝置喚醒。

Q8 : Channel Prepare

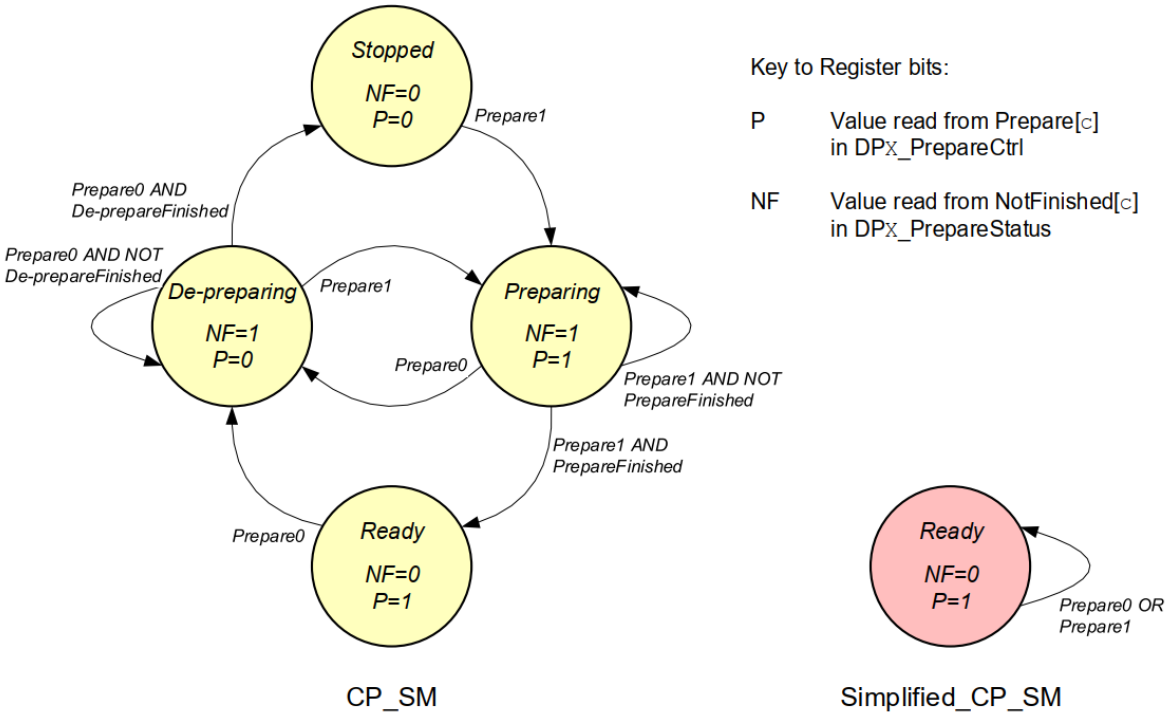


Figure 141 Channel Prepare State Machine (CP_SM)

Table 115 Stimulus to the Channel Prepare State machine (CP_SM)

Stimulus Name	Description	Notes (informative)
Prepare1	The most recent value written to the Prepare[c] bit in the DPX_PrepareCtrl Register is 1 *(see Figure 89 and Figure 106)*	Read-write register bit in CP_SM. Read-only / write-ignored bit for Simplified_CP_SM.
Prepare0	The most recent value written to the Prepare[c] bit in the DPX_PrepareCtrl Register is 0 *(see Figure 89 and Figure 106)*	Read-write register bit in CP_SM. Read-only / write-ignored bit for Simplified_CP_SM.
Prepare_Finished	The internal operations associated with preparing Channel c have completed	The internal operations are implementation-defined. There might be no operations so that the Prepare completes instantly.
De-prepare_Finished	The internal operations associated with de-preparing Channel c have completed	The internal operations are implementation-defined. There might be no operations so that the De-prepare completes instantly.

5. {4705} Each CP_SM shall change state in response to the stimuli as shown in Table 116.

Table 116 State Transitions in the Channel Prepare State machine (CP_SM)

Current State	Stimulus Value	Next State	Notes (informative)
CP_Stopped	Prepare1	CP_Preparing	
CP_Preparing	Prepare1 AND Prepare_Finished	CP_Ready	
CP_Ready	Prepare0	CP_De-preparing	
CP_De-preparing	Prepare0 AND De-prepare_Finished	CP_Stopped	
CP_Preparing	Prepare0	CP_De-preparing	Prepare operation was aborted before it completed
CP_De-preparing	Prepare1	CP_Preparing	De-prepare operation was aborted before it completed

Table 117 Effect of Channel Prepare State Machine on Active Channels

ChannelEn[c] bit in DPn_ChannelEn Register	State of CP_SM[c] or Simplified_CP_SM[c]	State of CP_SM[c] when ChannelEn[c] changed 0 to 1	Channel Transport Behavior	Notes (informative)
0	—	—	Idle – no transport	Channel is disabled
1	Simplified_CP_SM = CP_Ready	—	Active – transporting valid Samples	Channel was enabled
1	CP_SM = CP_Ready	CP_Ready	Active – transporting valid Samples	Channel was enabled after being properly prepared
1	CP_SM = CP_Ready	CP_Preparing CP_De-preparing CP_Stopped	Active – transporting implementation-defined Data (See permissions 5 & 6)	Channel was enabled before preparation was finished (an incorrect programming sequence)
1	CP_Preparing CP_De-preparing CP_Stopped	—	Active – transporting implementation-defined Data (See permissions 5 & 6)	Channel was enabled before preparation was finished, or is being de-prepared (an incorrect programming sequence)

Step by Step

Case 1: BUS上沒有Active channel, 且CurrentBank = Bank0

6. 設定Bank0 FrameShape並切換成Bank0 (Write PCP_FrameCtrl0)
7. 設定Payload sampling等Payload 相關Parameter (both Bank0 & Bank1)
8. 設定預計要啟用的Channel (Write DPn_Channel1 Register in Bank1)
9. 預告Peripheral預計啟用的Channel，寫入Prepare bit (Write DPn_PrepareCtrl register, not Banked)
10. 確認是否Prepare Finish(N-Finished ChannelX == 0 in DPn_PrepareStatus, not Banked)
    a. 方法1: Manager可藉由提前設定中斷Mask (Port Ready DPn_IntMask[1])，待所有de-prepare or prepare finish發PREQ。Manager再進行讀取確認狀態
    b. 方法2: Manager polling PrepareStatus NotFinished bit

11. 設定 PCP_FrameCtrl1，即切換至Bank1並於下一個Frame開始串送對應Channel Stream

Case 2: BUS上已有Active channel 且CurrentBank = Bank1，僅想要修改DP1的Active channel

1. 設定Payload sampling等Payload 相關Parameter (in Bank0)
2. 設定預計要啟用的Channel X (Write DP1_ChannelX Register in Bank0)
3. 設定預計要關閉的Channe Y (Write DP1_ChannelX Register in Bank1)
    a. 儘管處於CP_SM=CP_Ready 此刻Channel Transport皆會變成Idle
4. 預告Peripheral想啟用/關閉的Channel，寫入Prepare bit (Write DP1_PrepareCtrl register, not Banked)
5. 確認是否Prepare Finish(N-Finished ChannelX == 0 in DP1_PrepareStatus, not Banked)
    a. 方法1: Manager可藉由提前設定中斷Mask (Port Ready DP1_IntMask[1])，待所有de-prepare or prepare finish發PREQ。Manager再進行讀取確認狀態
    b. 方法2: Manager polling PrepareStatus NotFinished bit
6. 設定DP0, 2~15 NextInvertBank = 1，待會切換Bank0能繼承現有Bank1已有之設定
7. 設定DP1 NextInvertBank = 0，待會切換Bank0能套用新的Bank0設定
8. 設定 PCP_FrameCtrl0，即切換至Bank0並於下一個Frame開始串送對應Channel Stream

Note:

1. 如果強制設定Current Bank DPn_ChannelEn，會造成Sample Interval counter不對齊。因為這操作並不會發出SSP

Q9：SSP 的作用是什麼?

Payload transport and sampling Event 特性:

1. Payload 的傳輸和取樣是通過 Data Port 內部的 Counter 測量 bitslot interval 來進行的。為了保證傳輸的正確性，傳輸端與接收端的 Data Port，兩端的 Counter 必須要同相。
2. 不同的 data port 可能會用不同的 sample interval 在運作，但因為同相，data port 會定期出現所有 counter = 0，其週期為每個Data Port Sample Interval的最小公倍數

為了達成以上特性，使用了Stream Synchronization Points (SSPs)機制來強制每個 sample interval 都要同相。當 SSP發生時，所有的 sample interval counters 都會強制設為 0

Stream Synchronization Points (SSPs) 可以藉由以下方法產生:

1. 使用 Ping Control Word，並且將 SSP bit (bit 5) 設成 1
2. 寫入 PCP_FrameCtrl0 or PCP_FrameCtrl1 Register

Stream Synchronization Points (SSPs) 發起時機限制:

1. 如果系統上有超過一組Active Channel，SSP必須與前次SSP間隔 **DPs Sample Interval** 公倍數
2. 如果系統上沒有任何Active Channel，SSP可以於任意時間點發出。

Note:

1. SSP出現在 Frame 的邊界，Frame[N] BitSlot[MaxRol, MaxCol])結束時的Falling Edge of Clock (也就是 Frame[N+1] BitSlot[0, 0] 開始時的Falling Edge of Clock)
2. sample interval counter收到 SSP後會 強制歸零 (SSP至少要100ms以內發一次)

> {3001} When there are one or more Data Ports in the system transporting a Payload Stream from an active channel, the Manager shall make an attempt to indicate an SSP at least once every 100 milliseconds using one or other of the following methods:
>
> A. Generating a Ping Command with a value of 1 in the SSP field
> B. Generating a Write Command to the PCP_FrameCtrl register in all Peripherals that are currently Attached (see also Permission 1 in section 9.2.3).

Q10 : SoundWire 有哪些 Reset?

SdW Reset 分類

SoundWire 有三種 reset 都會讓 Device 狀態轉為 `Unattached`, 從而需要重新 attached 到 bus 來等待列舉 (enumeration), 下面根據對系統的影響從小排到大, 這三種 reset 分別是:

1. **Soft Reset**
    - Soft Reset 一般發生在設備發生同步錯誤的時候, 當設備同步碼出現太多錯誤 bits 或者連續 2 Frames 錯 1-bit 時就會發出 Soft Reset, 這種 reset 對設備的影響最小, 僅僅是用來防止設備在錯誤的時間往 bus 發送數據。當設備重新 attach 上 bus 後並且完成列舉之後, Master 就可以讀取設備的 register 來進行 debug
    - Soft Reset 後, Device Register 值應該都還在 (不會被清掉)
2. **Hard Reset**
    - 當發生 `clock stop` 或是往 0x0044 `PCP_Ctrl` 暫存器寫 `ForceReset` 欄位時就會觸發 Hard Reset。Hard Reset 會對 Slave 內部多個(或全部)狀態有影響, 影響程度取決於 designer 怎麼設計
    - 當Peripheral從ClockStopMode1喚醒, 會發生ClockStopMode1Reset
3. **Severe Reset**
    - 對設備影響最大的就是 Severe Reset, 例如發生 POR 或 Bus Reset 時, 其影響範圍也涵蓋了 Hard Reset 影響的部分。甚至是一些從初始化之後就不會更改的部分 (例如 PHY Output Control) 也會被 Severe Reset 影響到
    - PHY Output Control : `SlewTime_Ctrl`, `DriveStrength_Ctrl`, 和 `CapLoad_Ctrl`

Power-On Reset (POR)

Master & Slave Device 上電時都會有此過程, 透過自己的 Hardware Reset Signal 將自身 Register 設定到 Default 狀態。而 Slave 在初次連接到 Bus 上 POR 後, 也會把自己定在 `Unattached` 狀態, 等待跟 Manager 做同步以進到 `Attached` 狀態, 還有後續繼續做列舉。

Bus Reset

如 Figure 30 所示, 如果設備連續接收了超過 4096 個 Logic 1 時 (也就是 2048 個 clock cycle), 此時就會在第 4096 個 Logic 1 時觸發 Bus Reset。而之後只要保持發送 Logic 1, 則設備就會一直 keep 在 reset 狀態。

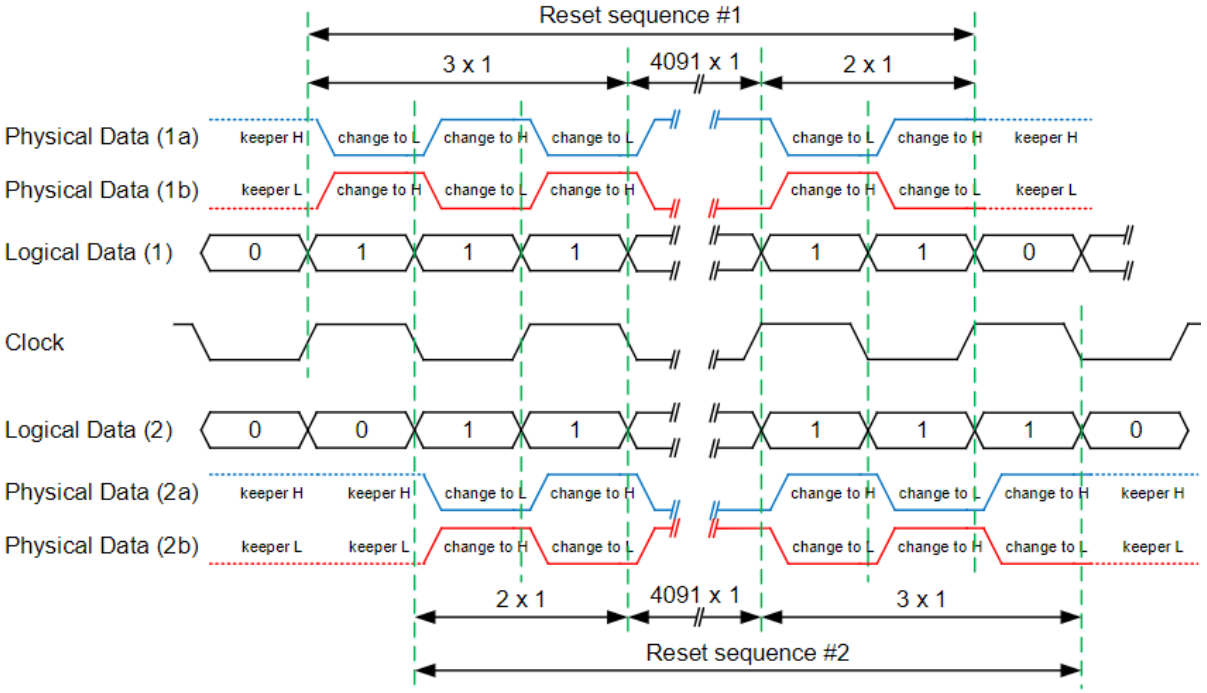


Figure 30 Bus Reset Waveforms

Q11 : Payload Transport 怎麼設定?

Sample Interval := separation of Sample Events measured in BitSlots.

Payload Data Interval := BlockGroupCount * Sample Interval.

Sample Window := Sample Interval number of consecutive BitSlots.

Payload Data Window := Payload Data Interval number of consecutive BitSlots.

Sample Window ⊆ Payload Data Window.

i.e., Sample Window is a subset of or equal to Payload Data Window.

Payload Channel Sample := SampleWordLength number of data bits.

Payload Data Container := ((0 or 2) + SampleWordLength) number of data bits.

Payload Data Sub-Block := 1 Payload Data Container.

Payload Data Block := (BlockGroupCount * NumChannels) number of Payload Data Containers

Transport Sub-Frame := all those BitSlots that are in columns from HStart to HStop inclusive.

Transport Sub-Frame ⊆ Frame.

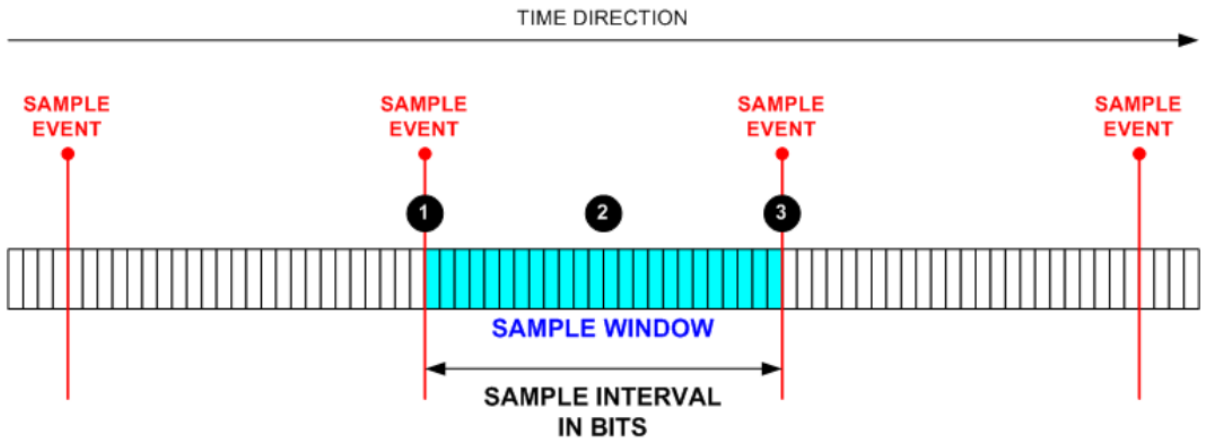
i.e., Transport Sub-Frame is a subset of or equal to Frame.

Payload Transport Window := Payload Data Window ∩ Transport Sub-Frame

i.e., the set of all BitSlots that are in both a given Payload Data Window and a Transport Sub-Frame.

Payload 相關名詞解釋

- **Sample Event**
  - 一筆取樣數據的第一個 bitslot
  - Sample Event 由 DP 產生，規格書沒有定義 Sample Event 要何時發生
- **Sample Interval**
  - 兩個 Sample Event 之間相隔的 bitslot 數量
  - 為了讓每個 Frame 內的 Sample Interval 為整數，通常 Frame Shape 要是 Sample Interval 的整數倍 (但其實沒有整數倍也可以 Work)
- **Sample Window**
  - 兩個 Sample Events 之間的 bitslots 集合



- **Payload Data Window**
  - 最多可以把 4 個 Sample Windows 組在一起，形成一個 Payload Data Window
  - BlockGroupCount = 1 ~ 4
- **Payload Data Interval**
  - 整個 Payload Data Window 內的總 BitSlot 數量

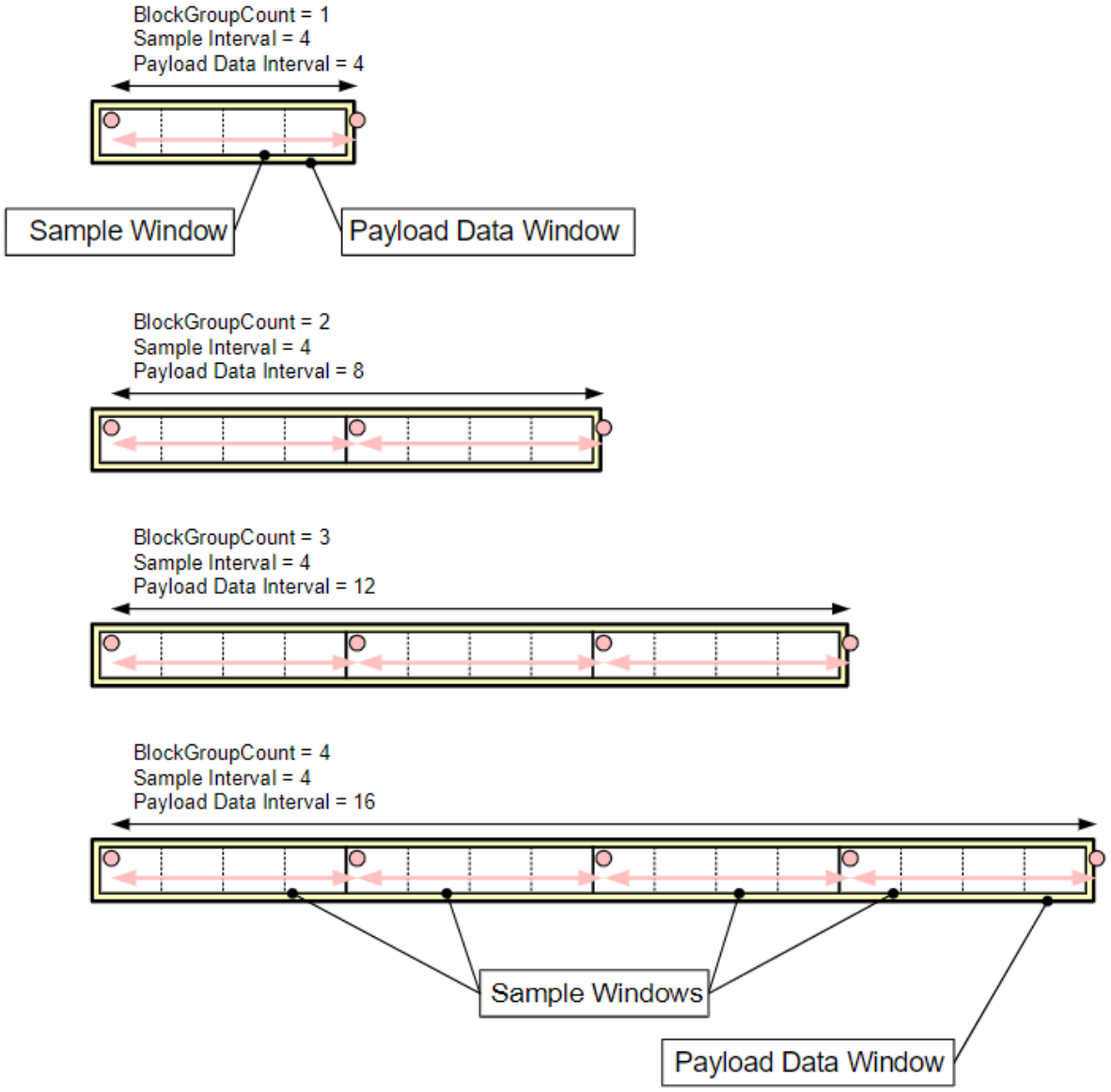


Figure 122 Grouping Sample Intervals to form a Payload Data Interval

- **Transport Sub-Frame**
  - 使用 `Hstart` & `Hstop` 參數來定義 Sub-Frame
  - 一個 Sub-Frame 內的 BitSlots 就是一個 DP 要傳輸的數據
  - 每個 DP 定義的 Sub-Frame 大小皆可不同，但不能相互重疊

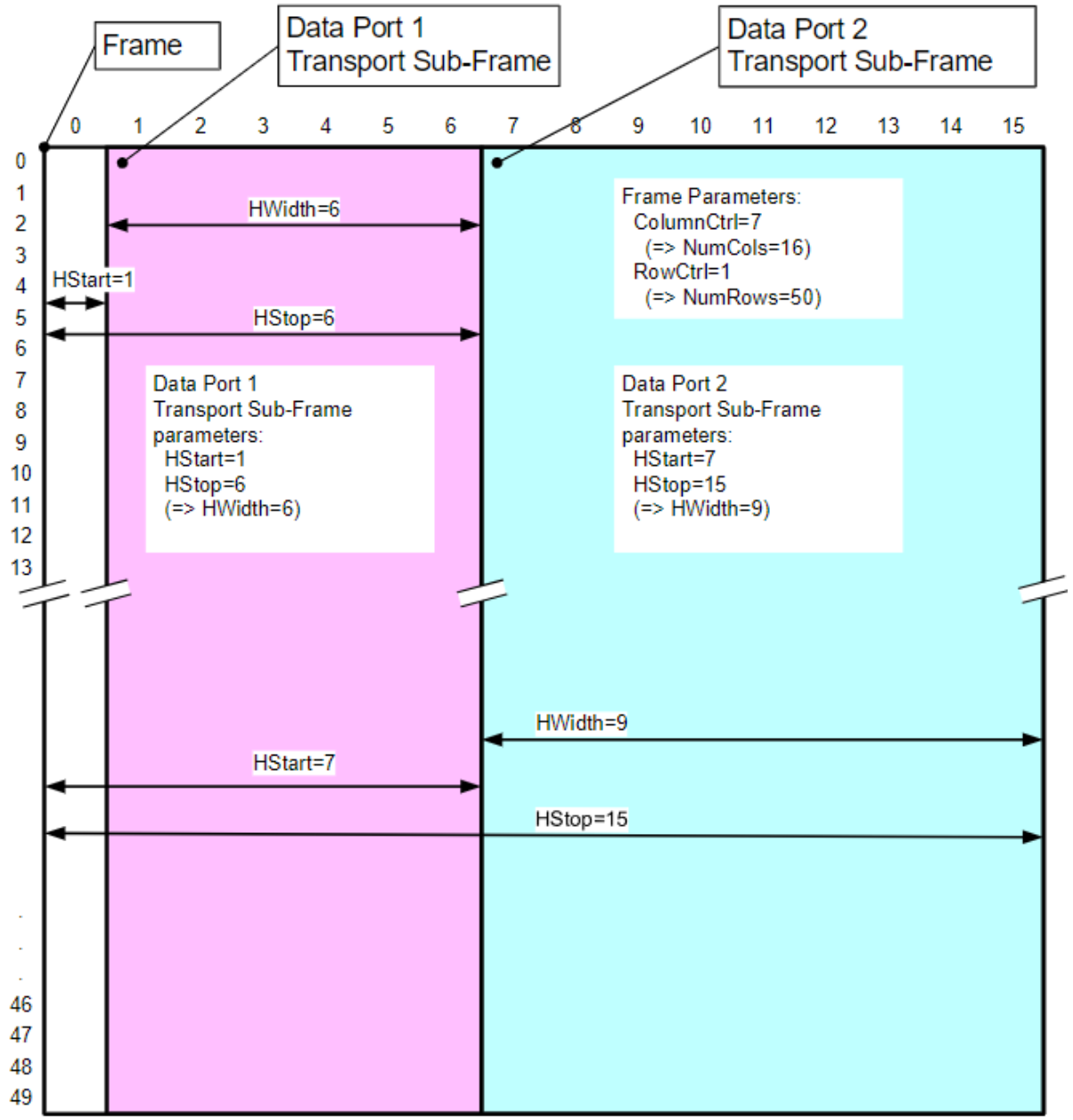


Figure 126 Transport Sub-Frame Examples

- **Payload Transport Window**
  - 為 Payload Data Window 與 Transport Sub-Frame 的交集區域
  - DP 會使用 Payload Transport Window 來區分哪些 BitSlots 屬於哪個 Stream

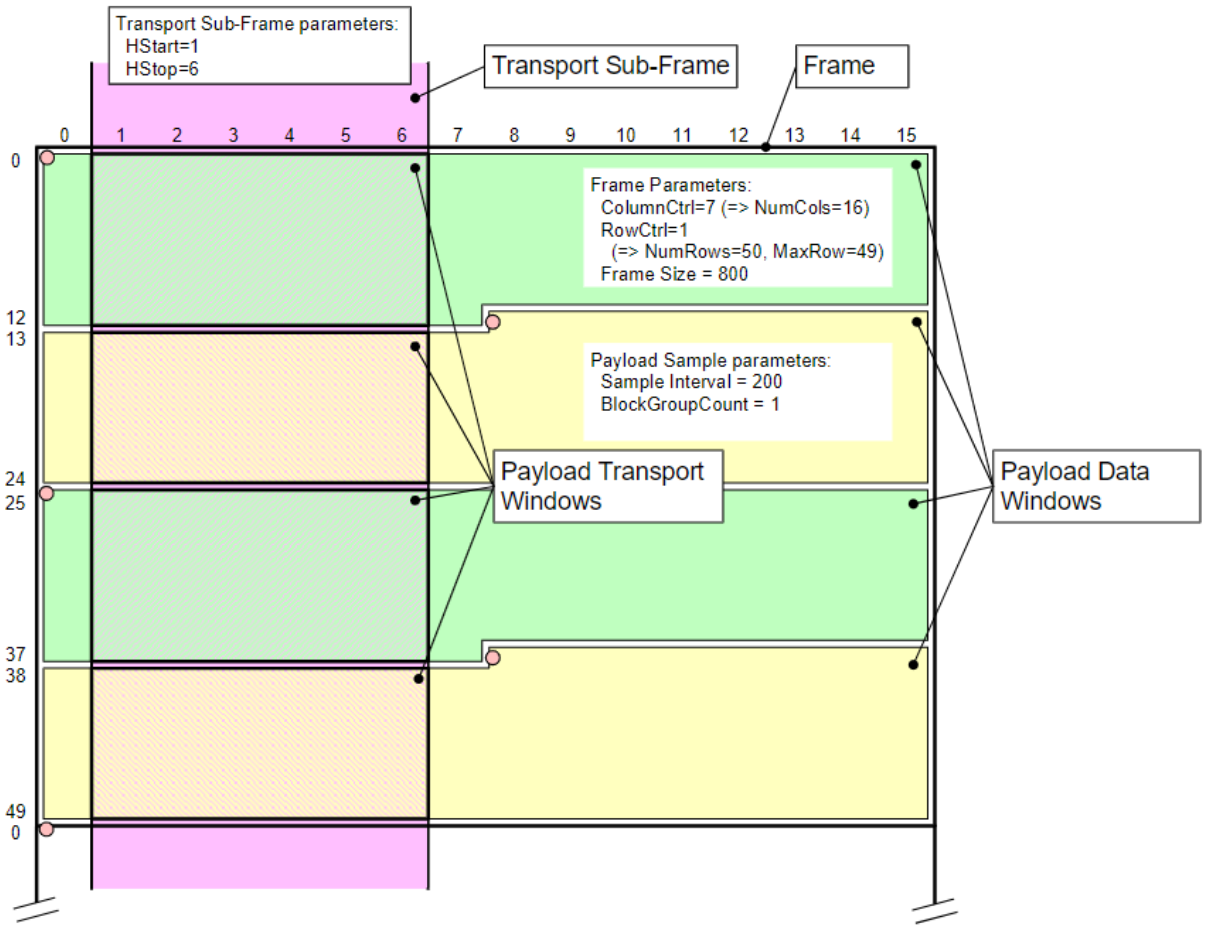


Figure 128 Payload Transport Windows Example

- **Payload Data Container**
  - 是傳輸 audio sample 的最小單位
  - `WordLength` 參數決定 Payload Data Container 的長度 (1~64)
  - Payload Data Container 的 data order 為 big endian，且空白處填充 0

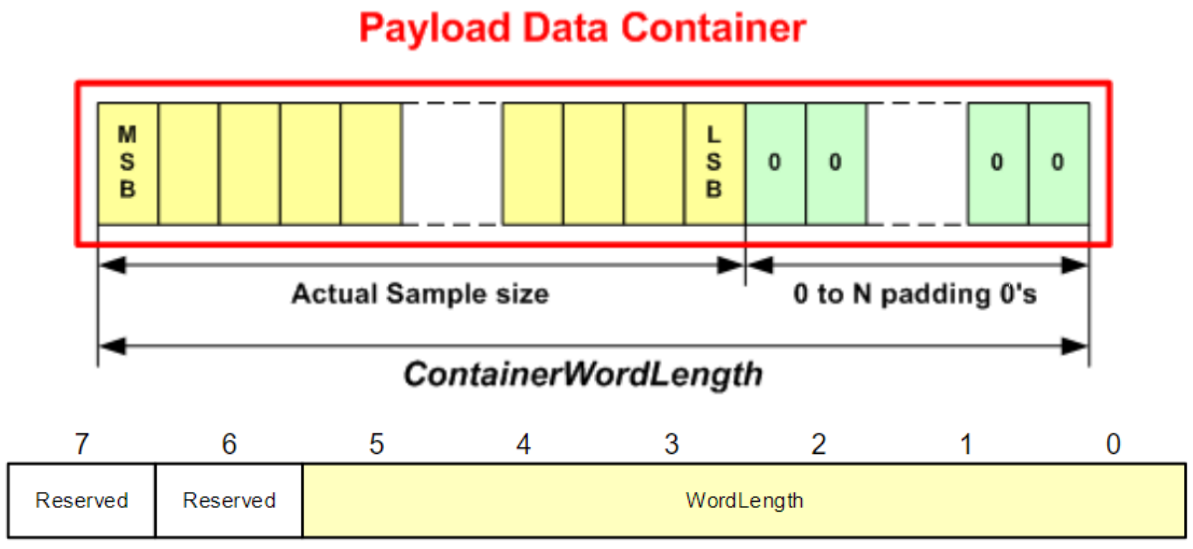


Figure 87 DPN_BlockCtrl1 Register

- **Payload Data Block**
  - 由一個或多個 Payload Data Container 所組成
  - 是 DP 進行傳輸的最小單位
  - 有兩種模式可以選擇：`BlockPackingMode = Block-per-Port Mode / Block-per-Channel Mode`

Payload 可設定參數總結

1. **BlockGroupCount**
   a. 範圍為 1 ~ 4
2. **BlockPackingMode**
   a. Block-per-Port Mode
   b. Block-per-Channel
3. **Hstart & Hstop**
   a. Hstart : Sub-Frame 從第幾行開始
   b. Hstop : Sub-Frame 共有幾行
4. **Offset**
   a. BlockOffset for Block-per-Port Mode
   b. SubBlockOffset for Block-per-Channel Mode
5. **Sample Interval**
6. **PortDataMode**
   a. Normal
   b. PRBS_Test
   c. Static_0
   d. Static_1
7. **PortFlowMode**
   a. Normal
   b. TX_Controlled
   c. RX_Controlled
   d. Full_Asynchronous
8. **WordLength**
   a. 範圍為 1~64

Q12 : Interrupt 有沒有什麼限制

(Hint : 例如多久要看到 PREQ，如果連續兩個 interrupt 來，host 如何知道？)

Interrupt 流程

1. New Event : Peripheral 有未處理的 Interrupt
2. Peripheral assert `PREQ` in Command Field
3. Manager 看到 `PREQ`，並發出 Ping Command 蒐集所有 Peripheral 的 `Per_Stat_NN`
    a. 當 Peripheral 已經因為有新的 `Alert` 發出 `PREQ`，直到收到成功的Ping command 之前會一直發出PREQ
    b. 從 `ClockStopMode0` 重新回到 Attached 時，如仍有Interrupt Staus未清除，會再次發出 `PREQ`
4. Manager 詢問有 `Alert` status 的 Peripheral (速度由 System defined，自行定義優先順序)
5. Manager 讀取 `PCP_IntStat` 來確認有哪些 Actived & Enabled Int (找出哪個 DP 有 Interrupt)
6. Manager 讀取有 Interrupt 之 DP 的 `DPx_IntStat` (速度由 System defined，自行定義優先順序)

    注意 `DPx_IntStat` 若沒有經過 `IntMask (IntState & IntMask!=1)`，讀取者須要自行處理

7. Manager 處理對應的 `IntStat` signal (System defined 自訂優先順序)
8. Manager 處理好對應的 `IntStat` signal，並且將 `DPx_IntClear` 寫入 1，以清除對應 Interrupt
9. 回到 Step6，依序清除該 Data Port 的每個 Interrupt，直到該 DP Interrupt 皆清除
10. 回到 Step5，確認還有哪些 Data Port 有 Interrupt，直到所有 DP Interrupt 皆清除
11. 回到 Step2，確認還有哪些 Peripheral 還有 Alert，並繼續處理 Interrupt 直到所有 Peripheral 皆沒有 Interrupt

Note:

1. 需特別注意，**DataPort0** 額外多了 **SDCA_Cascade Interrupt**
2. 列舉前不建議打開 **Interrupt (Mask)**
    a. 不可有效的讀取，因為可能有多個 **Peripheral** 同時回應
    b. 不可有效的發出 **Interrupt alert**
3. **PortN Cascade** 皆有經過 **Mask**，而 **DPn_IntStat** 未經過 **Mask**

Peripheral Status values were ever different, e.g. Alert and Attached_OK, then this would generate a Command Bus Clash condition, resulting in the Ping Command being aborted and the Peripheral Status ignored by the Manager.

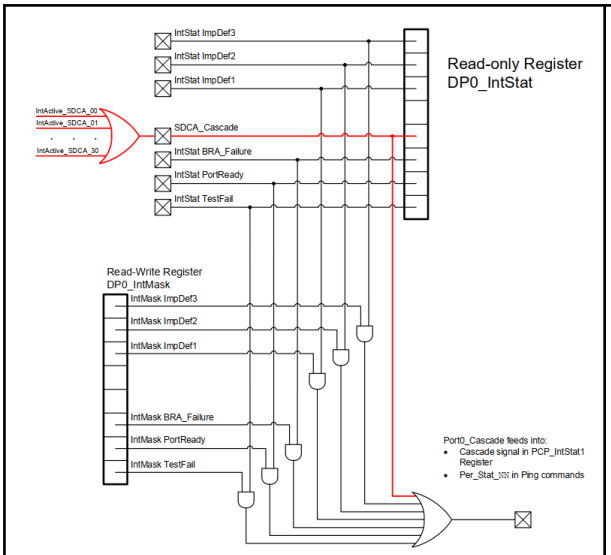


Figure 118 Cascaded Interrupt Status in Peripheral Data Port 0

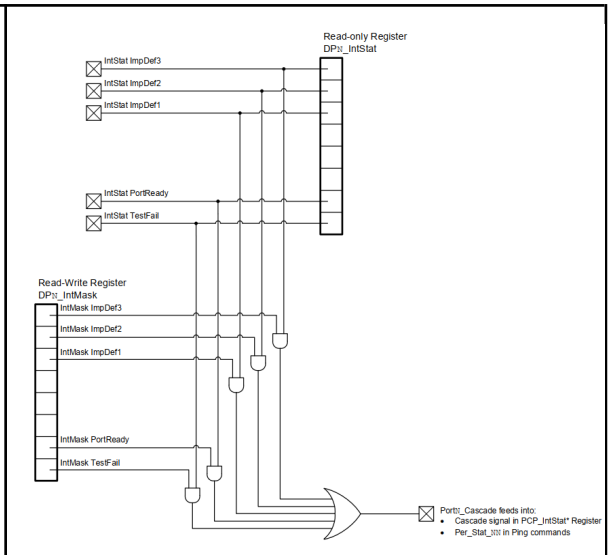


Figure 117 Cascaded Interrupt Status in a Peripheral Data Port (1–14)

0x0040	PCP_IntStat_1	$ PCP2 cascade	$ Port 3 cascade	$ Port 2 cascade	$ Port 1 cascade	$ Port 0 cascade	*$ IntStat ImpDef1*	$ IntStat Bus Clash	$ IntStat Parity
0x0040	PCP_IntClear_1	—	—	—	—	—	** IntClear ImpDef1*	* IntClear Bus Clash	* IntClear Parity
0x0041	PCP_IntMask_1	—	—	—	—	—	*IntMask ImpDef1*	IntMask Bus Clash	IntMask Parity
0x0042	PCP_IntStat_2	$ PCP3 cascade	$ Port 10 cascade	$ Port 9 cascade	$ Port 8 cascade	$ Port 7 cascade	$ Port 6 cascade	$ Port 5 cascade	$ Port 4 cascade
0x0043	PCP_IntStat_3	—	—	—	—	$ Port 14 cascade	$ Port 13 cascade	$ Port 12 cascade	$ Port 11 cascade

上圖為 Interrupt Stat bit 繼承關係，如下

- PCP3 cascade bit = (Port14 cascade) OR (Port13 cascade) … OR (Port11 cascade)
- PCP2 cascade bit = (PCP3 cascade) OR (Port10 cascade) … OR (Port4 cascade)
- PCP1 cascade bit = (PCP2 cascade) OR (Port3 cascade) … OR (Port0 cascade)

Q13 : BRA 是什麼，要怎麼使用?

(Hint : 就 protocol 上面與 state machine 來解釋)

BPT (Bulk Payload Transport) 用來是傳輸 Payload Channel Sample 的協定，而 BRA 則是 BPT 協定中定義的一種 Payload Type，可以在 Table 120 (`0x0002 DP0_PortCtrl` 暫存器 Figure 103) 的 `BPT_PayloadType` 欄位中選擇：

Table 120 BPT_PayloadType

BPT_PayloadType (see DP0_PortCtrl register in Figure 103)	Payload Type	Description
b00	Bulk Register Access	(see 13.2.2)
b01	Implementation-defined	(see 13.2.1 requirements 8 and 9 and permissions 2 and 3)
b10	Reserved	(see 13.2.1 requirement 7 and recommendation 1)
b11	Reserved	(see 13.2.1 requirement 7 and recommendation 1)

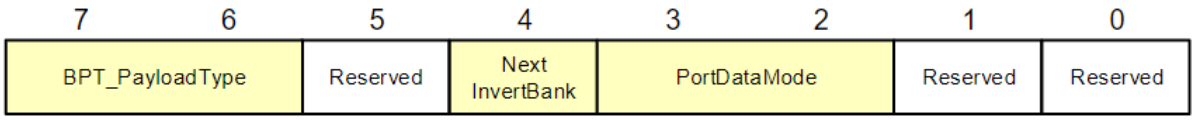


Figure 103 DP0_PortCtrl Register

BRA (Bulk Register Access) 是一種傳輸頻寬比一般 Read/Write Command 還要大的讀寫機制，可以在一個 Frame 中存取 1 ~ 502 bytes 的數據，並且每個 Frame 只能傳一個 BRA Block。

但要注意使用 Data[0] (Data Lane 0) 時單一個 Frame 最大只能 access `470 bytes`，只有用 Data[1] ~ Data[7] (Data Lane 1~7) 才可以 access 到最大 `502 bytes`。

BRA 會用 BPT Payload Stream Abstraction 把一個 Frame 裡的 Payload Channel Samples 的位元合併到單一個 BRA Block，然後將其視為一組 bytes。

Requirements of BPT

若要用 DP0 做 BPT 傳輸，需要以下設定：

- DP0 的 `PortFlowMode` 要固定在 Normal (Isochronous)
- DP0 要 enable channel
  - DP0 本身最多只支援一個 channel
- DP0 的 `BlockPackingMode` 要固定在 `block-per-port` Mode
  - DP0 只有一個 channel，所以永遠也只會用 `block-per-port` Mode
- DP0 的 `BlockGroupCount` 要固定為 1
  - DP0 在多個 Payload Data Blocks 中只有一個 Block 會有一個 Payload Data Container，所以不需要 `BlockGroupControl`

BRA Blcok Structure

- BRA Header
    - Required operation (Idle, Read, Write)
    - Address of device (Device Number)
    - Number of bytes (Actual 470 bytes or 502 bytes)
    - Register Address (4 bytes)
- BRA Header CRC
- BRA Header Response

  (用於確認 Target(s) 是否接受) - Command_Fail/OK/Ignored

- BRA DataBlock
- BRA DataBlock CRC
- BRA FooterResponse

  (用於確認 Target(s) 是否成功完成 DataBlock 操作)

Table 118 為 BRA Block 的範例:

Table 118 Example BRA Block

Byte Number	Byte Name	Source	Value	Meaning
0	BRA_Header[0]	Initiator	0xCA = b11001010	b11=> this Header is Active b0010 => Device Number 2 is selected b1 => Write operation b0 => MSB of BRA_NumBytes is 0.
1	BRA_Header[1]	Initiator	0x03	LSBs of BRA_NumBytes are 0x03 (so DataBlock is a total of 3 bytes)
2	BRA_Header[2]	Initiator	0x12	BRA_RegAddr[3] = 0x12
3	BRA_Header[3]	Initiator	0x34	BRA_RegAddr[2] = 0x34
4	BRA_Header[4]	Initiator	0x56	BRA_RegAddr[1] = 0x56
5	BRA_Header[5]	Initiator	0x78	BRA_RegAddr[0] = 0x78 (so Register Address is 0x12345678)
6	BRA_HeaderCRC	Initiator	0x19 = b00011001	Byte values are transported MSB first. When viewed in Figure 143 where the MSB (i.e., X^8 or Q[1]) is shown on the right, this value is mirrored, so corresponds to a value of $X^1 \ldots X^8$ (= Q[8:1]) = b10011000 = 0x98.
7	BRA_HeaderResponse	Target 2 (and all Targets)	0x08 = b00001000	Reading from MSB (transported first): b00 => Reserved field b0 => Target 2 is ready to perform a write b01 => OK (NAK=0, ACK=1) (b01 (OK) from Target 2 merged with b00 (Ignored) from all other Targets because their CRC check was successful) b000 => ImpDef response bits are 0.
8	BRA_DataBlock[0]	Initiator	0xED	Byte value 0xED is to be written to address 0x12345678 (the first address)
9	BRA_DataBlock[1]	Initiator	0xD1	Byte value 0xD1 is to be written to address 0x12345679
10	BRA_DataBlock[2]	Initiator	0xEA	Byte value 0xEA is to be written to address 0x1234567A

Byte Number	Byte Name	Source	Value	Meaning
11	BRA_DataBlockCRC	Initiator	0xD9 = b11011001	This is a write, so the DataBlockCRC comes from the Initiator. Byte values are transported MSB first. When viewed in Figure 143 where the MSB (i.e., X^8 or Q[1]) is shown on the right, this value is mirrored, so corresponds to a value of $X^1 \ldots X^8$ (= Q[8:1]) = b10011011 = 0x9B.
12	BRA_FooterResponse	Target 2	0x08 = b00001000	b00 => Reserved field b0 => Target 2 accepted the Write Data b01 => OK (NAK=0, ACK=1) b000 => ImpDef response bits are 0.
13 – n	(unused)	None	—	Any remaining bits in the BRA Block within this Frame are not driven by either Initiator or Target.

Note: **Bit order is bit endian** & **Byte order is lower address send first**

Table 122 為 BRA_Header 欄位解析:

Table 122 BRA_Header Fields

Byte and Bit index within BRA Header	Name	Notes (informative)
Byte[0] bits 7 – 6	BRA_HeaderType [25]	11 => Active 10 => Reserved 01 => Reserved 00 => Idle (Target ignores remaining bits of an idle BRA Block)
Byte[0] bits 5 – 2	BRA_DeviceAddress[3:0][25]	Selects one or more of the Target Devices *(see 13.2.3 Requirement 8)*
Byte[0] bit 1	BRA_Opcode	1 => Write, 0 => Read
Byte[0] bit 0	BRA_NumBytes[8] [26]	MSB of number of bytes to be read or written. Natural encoding so range is 0–511 (but in practice this is limited to 470 or 502 bytes by the limitations of Payload transport).
Byte[1] bits 7 – 0	BRA_NumBytes[7:0][25] [26]	LSBs of number of bytes to be read or written. Natural encoding so range is 0–511 (actually 470 or 502).
Byte[2]	BRA_RegAddr[31:24] [25] [26]	MSBs of address for first read or write operation
Byte[3]	BRA_RegAddr[23:16] [25] [26]	—
Byte[4]	BRA_RegAddr[15:8] [25] [26]	—
Byte[5]	BRA_RegAddr[7:0] [25] [26]	LSBs of address for first read or write operation

Table 123 為 BRA_HeaderResponse 的欄位解析:

Table 123 BRA_HeaderResponse Fields

Bit index within BRA Header Response	Name	Notes (informative)
7 – 6	Reserved	Target sends b00, Initiator ignores value
5	BRA_NotReady	0 => Target will process the Read or Write operation described by the Header. 1 => Target is not ready to process the Read or Write operation described by the Header, so will ignore the Read or Write operation and will not generate a BRA_FooterResponse.
4	BRA_HeaderNAK	NAK and ACK bits together form a 2-bit response field similar to the Response in the Control Word NAK+ACK values from Target: 00 => Ignored 01 => OK 10 => Failed (Header CRC check failed) 11 => Reserved NAK+ACK values received at Target or Initiator: 00 => Ignored 01 => OK 10 => Abort (Header cannot be trusted) 11 => Abort (Header cannot be trusted)
3	BRA_HeaderACK	part of 2-bit response field
2 – 0	Implementation-Defined[27]	—

Table 124 為 BRA_FooterResponse 的欄位解析：

Table 124 BRA_FooterResponse Fields

Bit index within BRA Footer Response	Name	Notes (informative)
7 – 6	Reserved	Target sends 0, Initiator ignores value
5	BRA_FooterResult	Writes: 0 => Good. Target accepted write payload 1 => Bad. Target did not accept write payload Reads: 0 => Good. Target completed read operation successfully 1 => Bad. Target failed to complete read operation successfully
4	BRA_FooterNAK	NAK and ACK bits together form a 2-bit response field similar to the Response in the Control Word NAK+ACK values from Target: 00 => Ignored 01 => OK 10 => Failed (DataBlock CRC check failed) 11 => Reserved NAK+ACK values received at Initiator: 00 => Ignored 01 => OK 10 => Abort (DataBlock cannot be trusted) 11 => Abort (DataBlock cannot be trusted)
3	BRA_FooterACK	part of 2-bit response field
2 – 0	Implementation-Defined[28]	—

BRA State Machine

Figure 144 為 Target 端的 BRA State Machine：

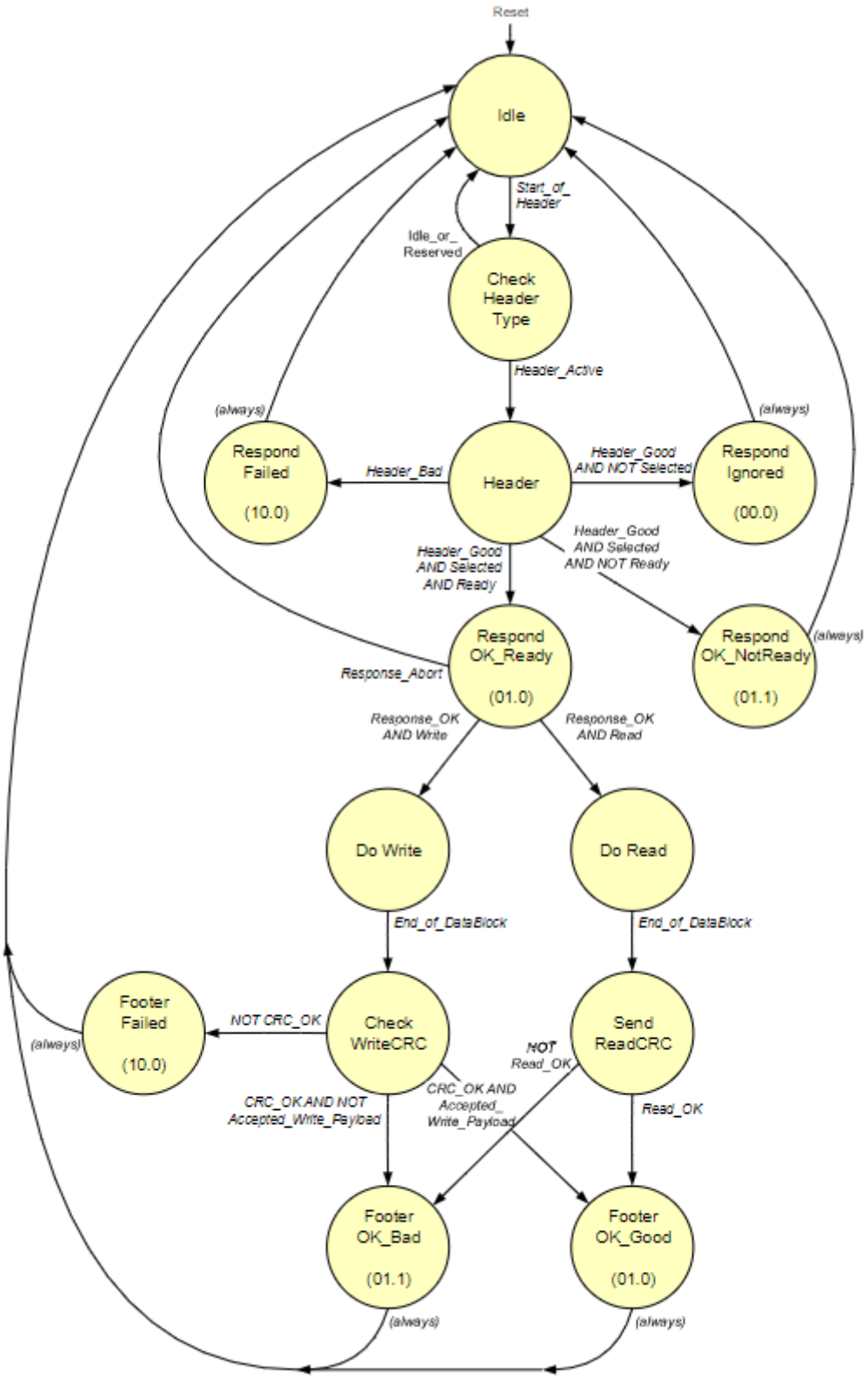


Figure 144 Bulk Register Access Target State Machine (BRAT_SM)

Figure 145 為 Initiator 端的 BRA State Machine

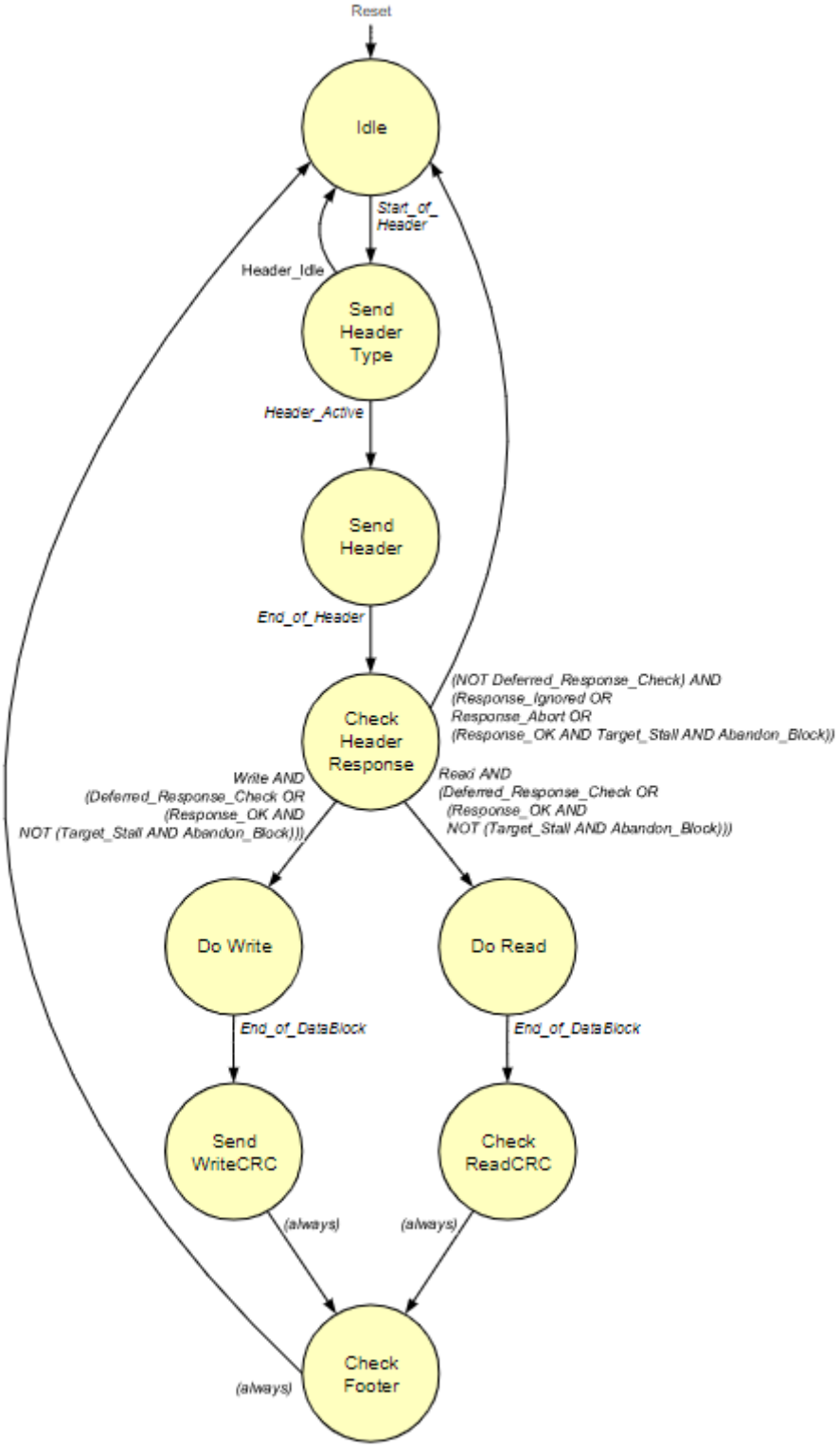


Figure 145 Bulk Register Access Initiator State Machine (BRAIn_SM)

Initiator 也可能會被設計成 batch mode，也就是它會先把所有的 `BRA_HeaderResponse`, `BRA_DataBlock` (for Reads), `BRA_FooterResponse` 都收進來，然後繼續發出下一個 `BRA_Header` (如果有的話)，接下來才會去檢查剛剛收進來的東西有沒有問題。

此外，假設接收端檢查 BRA_DataBlock CRC 發現 `BRA_DataBlock` 有錯誤時通常都為時已晚，因為可能已經有一部份數據被存放進暫存器了，這時可能會做的措施有以下：

- 繼續執行其本來打算執行的後續寫入操作組，並稍後再修復該組的問題
- 在後續 BRA Block 中發送 Idle 的 HeaderType 值，直到使用讀取命令或 BRA 讀取操作來詢問 Target，從而確定需要重新嘗試哪些操作

而如果 Initiator 看到 Footer Response 指示了 Abort，則 Initiator 要直接忽略 BRA 結果。