CFP2 Common Data Block

Application Note 5592

Introduction to CDB (Common Data Block)

When manufactured, each AFBR-842xxxx-xxx transceiver is loaded with default firmware that has proven performance and reliability. In general firmware update is not advised, however, if demand for deploying customized firmware exists, the CDB feature provides the option to do so via the MDIO interface without returning the transceiver to factory for rework. I.e. the transceiver's firmware can be changed when functional MDIO communication is established between the host and transceiver.

This application note is divided into six sections; CDB Structure, CDB Implementation, CDB Command, CDB Command Execution, CDB Command Execution Process and Firmware Upgrade

1. CDB Structure

CFP MSA MIS 2.2 r06a allocates 1024 registers on A000h page as the CDB which starting at AC00h and ending at AFFFh. It has two realizations, the CDB Command Frame (Panel A) and the CDB Reply Frame (Panel B). When host requests module to perform a task it writes a CDB Command Frame to the module. When host requests the status of last command execution with optional data return; it reads a CDB Reply Frame from the module. See Figure 13 Common Data Block Structure in CFP MSA MIS 2.2. CDB Command Frame includes CDB Command Register, CDB Payload Size Register, CDB Payload Registers and CDB CRC Registers.

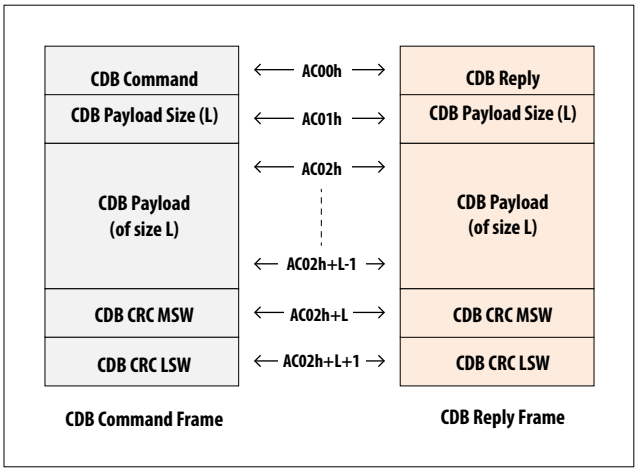

Figure 13.Common Data Block Structure

1.1. CDB Command Register

In a CDB Command Frame AC00h contains the CDB Command which is required to make a valid command frame. Host shall write to this register as the last register to write in a Command Frame. Writing to this register "triggers" the execution of the CDB command by module.

1.2. CDB Payload Size Register

CDB Payload Size register is at AC01h containing the total number of registers of CDB Payload starting at AC02h. The minimum value of CDB Payload Size is 1 and the maximum value is 1020. Zero value indicates that the payload is not present. Payload Size register shall be always present regardless of its value.

1.3. CDB Payload Registers

CDB Payload occupies a block of registers starting at AC02h and ending at AC02h+L-1, where L is the CDB Payload Size. CDB Payload contains parameters of other type of data associated with a CDB Command. CDB Payload is only present when CDB Command has data to pass the module.

1.4. CDB CRC Registers

The CRC registers are allocated right after the end of CDB Payload with MSW at smaller address. CRC is a 32-bit value with the same algorithm defined for B000h page Bulk Data Block. The use of CRC is optional, but when it is used, they always shall start at AC02h.

1.5. CDB Reply Frame

In a CDB Reply Frame AC00h contains the CDB Reply required to make a valid reply frame. Host shall read this register first to determine where it needs to read further for the rest of the Reply Frame. When preparing a Reply Frame, a module shall write to this register with "Command in Process" in the field of CDB Status indicating the rest of the Reply Frame may not be ready for host to read. The rest of a DCB Reply Frame is similar to that of a CDB Command Frame.

NOTE:
Figure and Table numbers in this document are referenced to:
CFP MSA Management Interface Specification
100/40 Gigabit Transceiver Package Multi-Source Agreement
Version 2.2 r06a
July 1, 2013
CFP_MSA_MIS_V2p2r06a.pdf

2. CDB Implementation

Please refer to Table 18 CDB Implementation in CFP MSA MIS 2.2 for the general form of CDB Command Frame and CDB Reply Frame with implementation details.

Table 18. CDB Implementation

Hex Addr	*Size*	*Access Type*	*Bit*	*Register Name*	*Description*	*Initial Value*
AC00	**1**			**CDB Command or CDB Reply**	**This is a shared address between CDB Command register and CDB Reply register, defined respectively with the following cases.**	**0000h**
		CDB Command Frame Case				
		WO	15~12	Reserved		0
		WO	11~8	CDB CMD Class	A 4-bit unsigned value coding 16 CDB Command Classes. 0h: System level operations 1h: General CFP register operations 2h~Dh: Reserved Eh~Fh: Vendor specific command class code.	0
		WO	7~0	CDB CMD Code	An 8-bit value coding 256 CDB commands for each CDB CMD Class. See separate section for details.	0
		CDB Reply Frame Case				
		RO	15~10	Reserved		0
		RO	9~8	CDB Status	A 2-bit value representing CDB status. Note this field shall be synchronized with A004h.3~2 therefore CDB and A004h can be operated one and only one at a time. 00b: CDB Idle, 01b: CDB Command completed successfully, 10b: CDB Command in progress, 11b: CDB Command failed.	0

			7~0	CDB Message	An 8-bit value coding CDB Message related to each CDB Status.
					If CDB Status = CDB Idle, then
					00h: Reserved,
					01h: Ready to accept host command,
					02h~7Fh: Reserved by MSA,
					80h~FFh: Allocated for vendor use.
					If CDB Status = Command in Progress, then
					00h: Reserved,
					01h: Command is captured but not processed,
					02h: Command checking is in progress, CDB Reply CRC is not valid.
					03h: Command execution is in progress,
					04h: Command execution is in progress but CDB Payload is open for host write, note that write to CDB Command register (AC00h) shall be ignored. Meanwhile, Payload Size and CRC in the Reply Frame may be overwritten by incoming host write.
					05h~7Fh: Reserved by MSA,
					80h~FFh: Allocated for vendor use.
						0

Continued on next page...

Table 18. CDB Implementation (Continued)

Hex Addr	*Size*	*Access Type*	*Bit*	*Register Name*	*Description*	*Initial Value*

					If CDB Status = CDB Command Completed Successfully, then
					00h: Reserved,
					01h: No specific message, one more host read gets CDB to idle status,
					02h~3F: Reserved by MSA,
					40h~7Fh: For individual CDB Command or task progress report ,
					80h~FFh: Allocated for vendor use.
					If CDB Status = Command Failed, then
					00h: Reserved,
					01h: CDB Data Length error, L > 1020,
					02h: Unknown command,
					03h: Command checking error without detail,
					04h: Command checking time out,
					05h: CRC error,
					06h: Password error,
					07h~0Fh: Reserved for CDB command checking error,
					10h: Command execution error without detail,
					11h~3Fh: Reserved by MSA
					40h~7Fh: For individual CDB command or task error,
					80h~FFh: Allocated for vendor use.	

AC01	**1**			**CDB Payload Size**	Contain the length of CDB Payload.	**0000h**
		RO	15~10	Reserved		0
		RW	9~0	Payload Size	A 10-bit unsigned integer L, 0= < L <= 1020.	0
AC02	**L**	**RW**	15~0	**CDB Payload**	**Data block of size L with either a CDB Command or Reply.**	**N/A**
AC02+ L	**2**	**RW**	15~0	**CDB CRC**	**32-bit CRC for the registers AC00h, AC01h, and CDB Payload. Most significant word at smaller address.**	**0000h**

3. CDB Command Execution

3.1. Initialization

On power up CDB function shall be initialized to CDB IDEL state with CRC and Global Alarm disabled.

3.2. Host to Write a CDB Command Frame

To send a CDB Command to module, host shall write CDB Command register AC00h at last and shall use this write as the "trigger" for module to execute the CDB Command. Host shall be able to write all other registers in a CDB Command Frame in any order. A module shall interpret other register contents per the CDB command once it detects the "trigger".

3.3. Host to Read a CDB Reply Frame

To received a CDB Reply Frame host shall read CDB Reply register (AC00h) first and then shall proceed with reading CDB Payload Size register and CDB Payload, and then CRC. If no data is return per the previous command, host only needs to read the CDB Reply register to determine the CDB Status and CDB Message.

3.4. Command in Progress (CIP)

COP is an important status for module to present to host. Once a command is received CDB state machine shall immediately update CDB reply register with this status and associated CDB Message. During this state, module shall not be able to determine its CRC content if CRC option is enabled. Host shall not make attempt to read CRC registers.

3.5. Command Completed Successfully (CCS)

CCS is asserted by a module with proper CDB Message for additional information. If CDB Payload is attached as a part of the CDB Command execution, host shall read the CDB Payload per CDB Payload Size register. If CRC is enabled, host shall read CRC registers as well to determine whether a valid CDB Reply Frame is valid.

3.6. Command Failed (CF)

CF is a CDB State indicating a failed execution of a CDB Command. The CDB Message 10 shall be used by a module to provide additional cause of failure.

4. CDB Command Execution Process

CDB Command execution is an interactive process between host, CDB State Machine, and module processor. Figure 2 CDB Command Execution Flowchart illustrates the process of host execution. Note that CDB Reply is used extensively in CDB Command execution process.

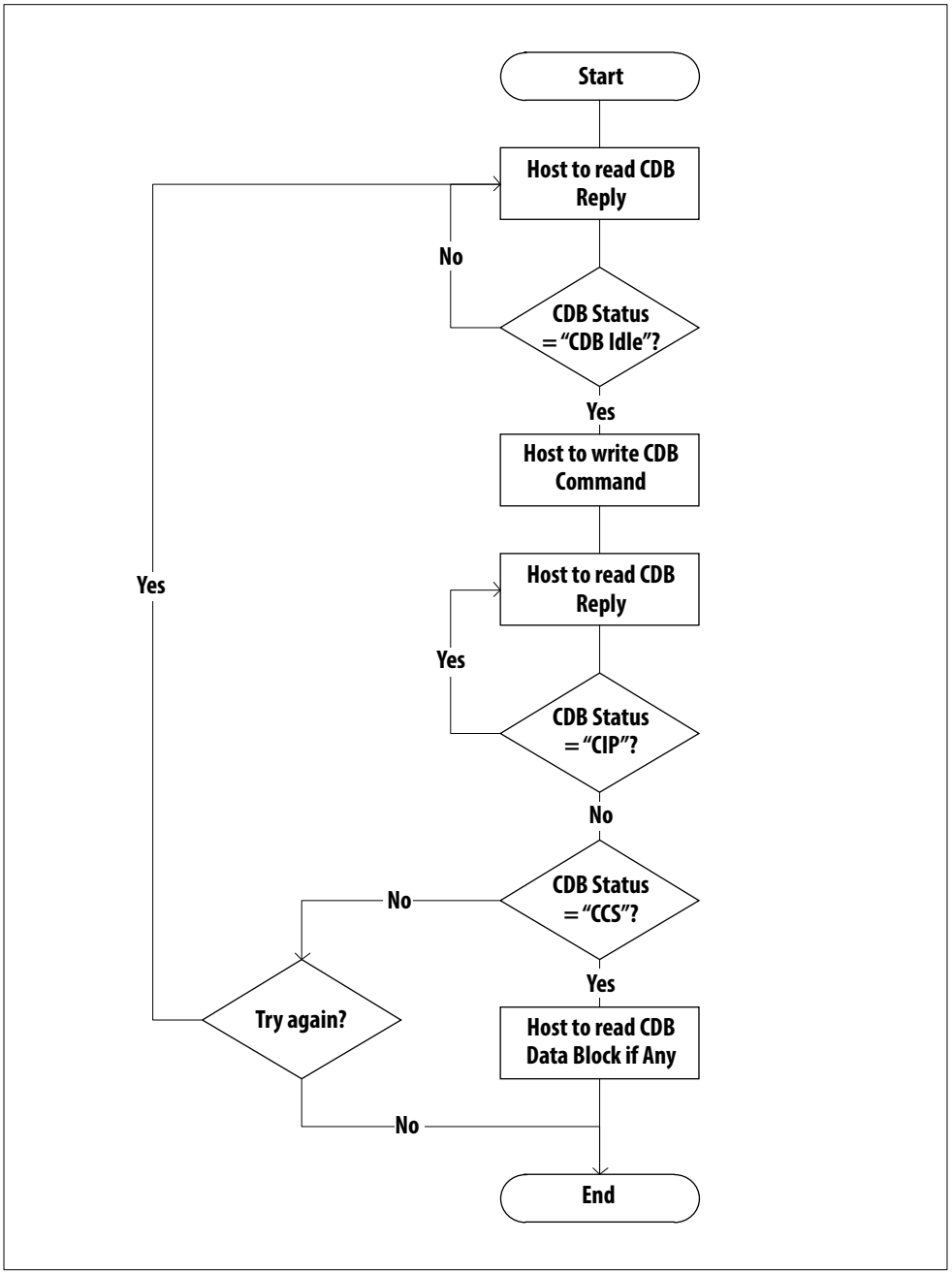

Figure 14. CDB Command Execution Flowchart

5. CDB Commands

CDB Commands are listed in Table 21 CDB Command Table where all the acronyms used are listed in Table 20 Acronyms used in CDB Command Table. AFBR-842XXXX-XXX supports CDB Command Class 0 – System Operation only. CDB Command Class 1 – Register Access is not supported

Table 20. Acronyms used in CDB Command Table

Acronym	Description
PW	Password
PS	Payload Size
PLx	Payload xth entry
Y	Yes or OK
N	No or None
CMD ID	Value in register AC00 in a CDB Command Frame. It is also used as a Command ID.

Table 21. CDB Command Table

CMD Class	CMD Code	CMD ID	Command Name	PS	PW	Description
CDB Command Class 0 – System Operation						
00h	00h	0000h	Reserved			
00h	01h	0001h	Commit Password	0	N	Commit to the password entered in A000h~A001h.
00h	02h	0002h	Save New Password	0	Y	Save the new password entered in A002h~A003h.
00h	03h	0003h	Enable Password	0	N	Enable the optional password protection for user NVR.
00h	04h	0004h	Disable Password	0	Y	Disable the optional password protection for user NVR.
00h	05h	0005h	Enable CDB CRC	0	N	Enable the optional CRC for CDB. This Command is volatile after power cycle. No CRC checking shall be executed on this command itself but CRC shall take effect starting from next CDB Command/Reply Frame if this command is executed successfully.
00h	06h	0006h	Disable CDB CRC	0	N	Disable the optional CRC for CDB. CRC checking shall be performed for this command itself but CRC shall be inactive starting from next CDB Command/Reply Frame.
00h	07h	0007h	Enable CDB Global Alarm Interrupt	0	N	Enable CDB Interrupt to host option. This Command is volatile.
00h	08h	0008h	Disable CDB Global Alarm Interrupt	0	N	Disable CDB Interrupt to host option.
00h	11h	0011h	Start Firmware Download	0	Y	Request the module to receive new firmware image with forth coming Bulk Data Write command. (Simulate B04Dh.15~12:1) Expected CMD specific Reply: 0140h: Ok to receive FW image, 0340h: Not enough NVM space, 0341h: Other errors.
00h	12h	0012h	Complete Firmware Download	0	Y	Request module FW image download is finished. (Simulate B04Dh.15~12: 2) Expected CMD specific Reply: 0140h: Full image has been received and image is good. 0340h: Image is incomplete, 0341h: Image is complete but CRC error,
00h	13h	0013h	Run Image A	0	Y	Request module to run Image A. (Simulate Bo4Dh.1~12: 3) Expected CMD Specific CDB Reply: 0140h: Command has been executed; 0340h: Image A is not valid, execution aborted. 0341h: other errors.

Continued on next page...

Table 21. CDB Command Table (Continued)

CMD Class	CMD Code	CMD ID	Command Name	PS	PW	Description
00h	14h	0014h	Run Image B	0	Y	Request module to run Image B. (Simulate B04Dh.15~12:4) Expected CMD Specific CDB Reply: 0140h: Command has been executed; 0340h: Image B is not valid, execution aborted. 0341h: other errors.
00h	15h	0015h	Abort Image Download	0	Y	Abort Image Download. (Simulate B04Dh.15~12:5). Expected CMD Specific CDB Replay: 0140h: Image download aborted. 0340h: Command error.
00h	16h	0016h	Copy Image A to B	0	Y	Request module to copy Image A to B (Optional) Expected CMD specific CDB Reply: 0140h: Command has been executed; 0340h: Not enough memory, execution aborted. 0341h: Copying is not successful, 0342h: Other errors.
00h	17h	0017h	Copy Image B to A	0	Y	Request module to copy Image B to A (Optional) Expected CMD specific CDB Reply: 0140h: Command has been executed; 0340h: Not enough memory, execution aborted. 0341h: Copying is not successful, 0342h: Other errors.
00h	18h	0018h	Commit Image A	0	Y	Request module to commit to Image A. Expected CMD Specific CDB Reply: 0140h: Committed to Image A. 0340h: Command error.
00h	19h	0019h	Commit Image B	0	Y	Request module to commit to Image B. Expected CMD Specific CDB Reply: 0140h: Committed to Image B, 0340h: Command error.
00h	20h	0020h	Get Software Upgrade Status	0	Y	Get Firmware Upgrade Status. Host issues this command to get CDB Reply with a CDB Reply payload of size 1. Expected Reply: 0140h: Status read successful 0340h: Status read error Payload PL0 takes the identical definition as register B051h. Note the following fields of B051h are not included: B051h.15~14.
00h	21h	0021h	Download Image Block	L	Y	Host to download a block of software image to module. PL0 = Image Block Number (max 65535). The rest of CDB Payload is the software image block which can contain additional descriptor per vendor design. Note total image size is limited to 1019 x 65536 = 66.78 MB. Expected CMD specific Reply: 0140h: Image download successful. PL0 = Block number just downloaded. 0340h: CRC image block CRC error. PL0 = Block number just downloaded.

6. Firmware Upgrade

Firmware upgrade procedure can refer to Figure 19 Software Upgrade State Machine and Figure 20 Software Upgrade Sequence with exception that copy commands are not supported.

For software update, the software data image must be divided into blocks whose size is determined by how much data can be processed the module in a given time cycle. Each block includes the data and CRC, so that the module can check whether there are any errors after receiving the block. Upon find any errors in the block, the module informs the Host for a received errored block and the host must retransmit the same block.

A software upgrade transfer begins with the Host issuing a request to download an image. The module grants the request and the image is written a block at a time and setting the "Upgrade Data Block Ready" flag and the module processes each block and updates the status. It is the host responsibility to make sure that each block size is equal to or less than the "Maximum Upgrade Data Block Size". If there is any error in block processing, the host will retransmit the block. It is recommended to force an abort by the host if a CRC error occurs few times on the same block. While download is not complete, the Host can issue "Abort" command to abort the current download that is in progress. After all the works of the image have been written to the module, termination of the transfer is completed by issuing a Download Complete to the Upgrade Command register. The module will acknowledge the complete image has been downloaded successfully by providing a Command completed successfully status. If the image has an error in download, then the module will reply with a Command failed status. This state machine is illustrated in Figure 19 Software Upgrade State Machine. The Software Upgrade sequence is illustrated in. Modules sets - Maximum Upgrade Data Block Size.

Once the image has been downloaded successfully, the image's service affectability will be reported and a request to run downloaded image can be performed. Ideally, most upgrades should not be service affecting, i.e. services actively supported by the transmission system, especially if they are just software upgrades. In some instances when upgrading firmware it may not be possible to achieve a non-service affecting upgrade. With the image service affecting status provided, the host software can be informed of the side effects that may impact current service by upgrading to the downloaded imaged. During a service affecting upgrade, the modules may be in a state where even MDIO transactions are not available to the module while the upgrade is happening. In order for the host to be cognizant of when MDIO transactions are available, the assertion of the GLB_ALRM pin shall signal to the host that initialization due to the upgrade is complete and the MDIO interface is available. Even though an upgrade is service affecting, it shouldn't require a reconfiguration of the module to get it in the operating state that it was in just prior to the upgrade.

After the run downloaded image request is issued by the Host, the module will be running the downloaded version of software. At this point, the Host can commit the image. If the Host wants to keep both banks the same, it can issue "Copy image" command.

Note: The host should be aware that during module software upgrade, the NVR Checksum may be inconsistent due to mismatch of some register values between host and module, e.g. 0x806Ch, 0x807Bh. These registers should be updated and the host, module NVR Checksums consistent after the module software upgrade is successfully completed.

To clarify expected host behavior following module hardware reset, there are cases that need to be considered:

1. Hardware Reset:

   Asserting MOD_RSTn will cause a complete reset of the module. All VR values are host and must be re-written by the host.

2. Non-service affecting upgrade

   Non-service affecting upgrades are typically software-only upgrades and will not include module reprogramming. If the VR is maintained in the module, the MDIO register space is preserved during the upgrade. The CPU must re-read the VR after the upgrade to return to the state prior to the upgrade. This will include channel numbers, power setting etc.

3. Service affecting upgrade

   Service affecting upgrades may include reprogramming of the module. During this process, the contents of the VR in the module may be lost and the host must reset the VR to return the module to the configuration state prior to the upgrade.

A default image is loaded in the both image A and B banks when the AFBR-842XXXX-XXX was manufactured. Executing this image in either bank restores the module to factory default state. This is useful when user want undo the software update and restore the transceiver to original factory default state. Password for User NVR1 & 2 will be restored to MSA default (0101 1100h) after restoring the module using the default image.

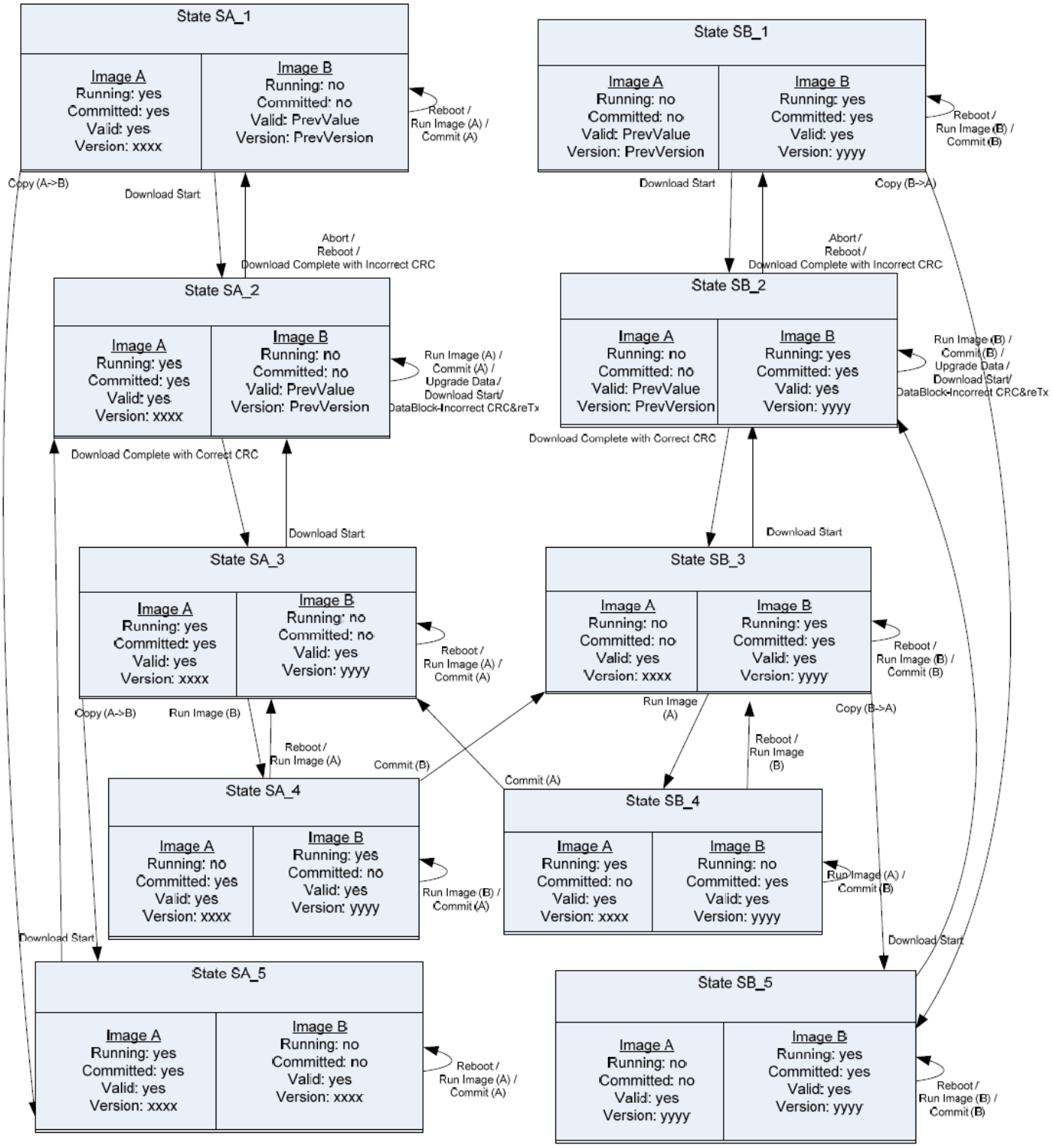

Figure 19. Software Upgrade State Machine.

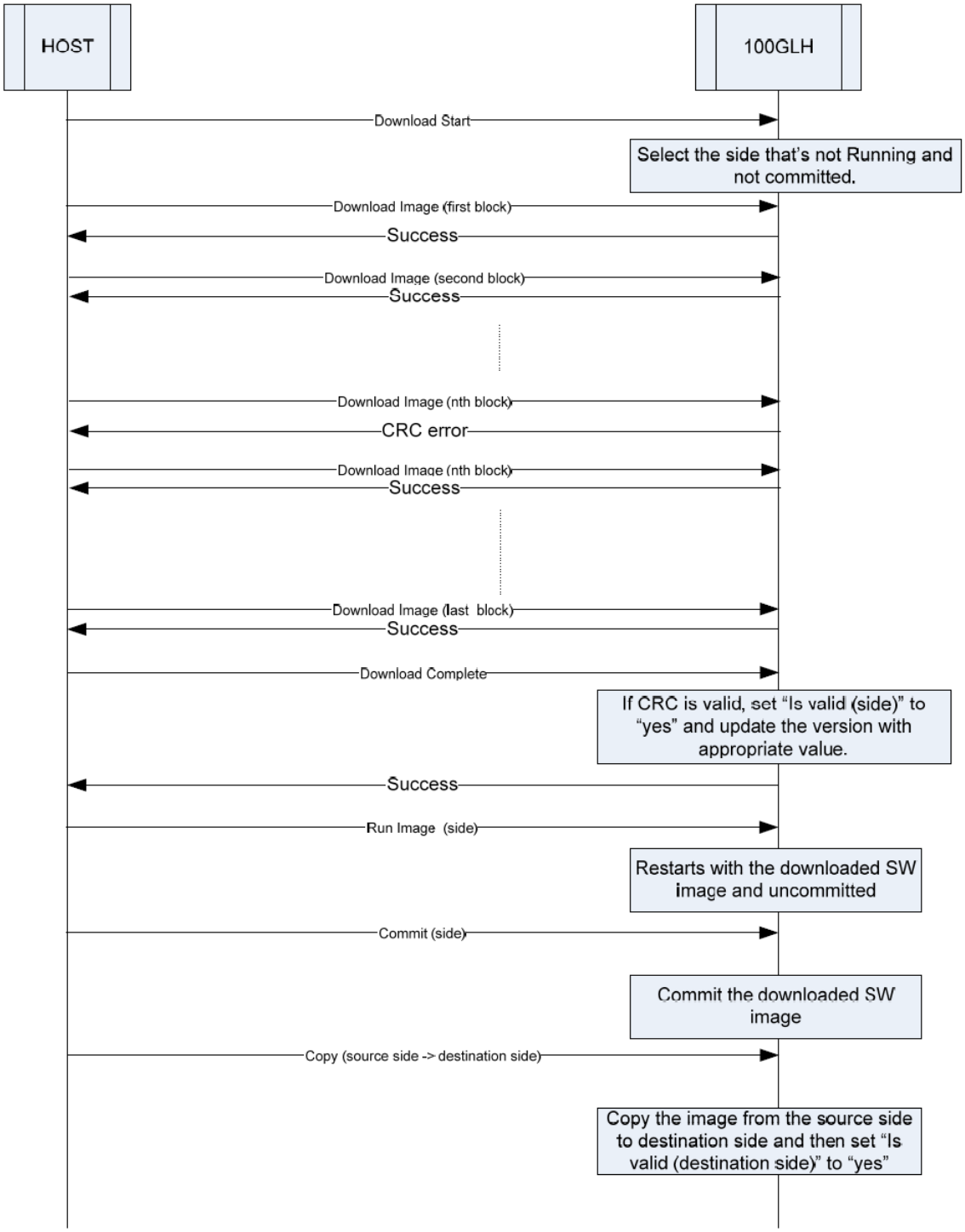

Figure 20. Software Upgrade Sequence

AV02-4766EN - February 16, 2015