CASN Page Introduction

Version: 1.0.2

Release date: 2024-08-21

Version History

Version	Date	Author	Description
1.0.0	2024-07-26	Sky Huang (SkyLake.Huang@mediatek.com)	First Release
1.0.1	2024-08-15	Sky Huang	Add example for ECC Parity Layout and Advanced ECC Transform mechanism
1.0.2	2024-08-21	Sky Huang	Fix description in Section 2.21 and 2.25

Table of Contents

1 Introduction

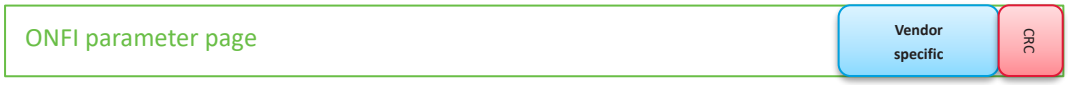
1.1 What is CASN?

CASN, i.e., **Common Attributes for SPI-NAND**, page is designed to fully describe SPI-NAND's parameters so that the host controller can address it and get correct ECC information from it. It has the same CRC mechanism as ONFI parameter page. However, CASN page is more advantageous on:

1. CASN page provides critical information like
    1) Flash model name
    2) Plane number
    3) Does the SPI-NAND device contain Quad enable bit or not
    4) Does the SPI-NAND device support continuous read or not
    5) Read/Write command set
2. CASN page contains flash on-chip ECC information by each flash vendor so that host driver can transform them into intuitive bitflip numbers and easily do wear-leveling.
3. We don't need to maintain SPI-NAND flash table in host driver anymore.
4. Accelerate compatibility test of SoCs and SPI-NAND flashes: You can easily replace your SPI-NAND with any other CASN compatible devices.

Each CASN page copy is 256 bytes long. There's three ways to integrate CASN page in current SPI-NAND design:

1. Add a brand-new command to read CASN page: With this, we can use a new 2KB or 4KB page to store CASN. However, there's too much effort because flash manufacturers need to re-design command set and find another flash memory array to store CASN data.
2. Use the "Vendor Specific" field of ONFI parameter page (byte 166~253): Space is not enough for CASN data.



3. Use the rest space besides ONFI parameter page:

From above, we know the best option will be option 3. Flash manufacturers can integrate CASN page easily if directly attaching it to the space after parameter page. As a result, host driver uses the same command(13h) to read CASN page as how it reads parameter page, just with a difference row address. Therefore, we set OTP-E bit in status register 2 (mostly B0h) first and then issue read command(03h) with CASN page column address(300h). Table 1 describes how CASN page's row address differs among manufacturers.

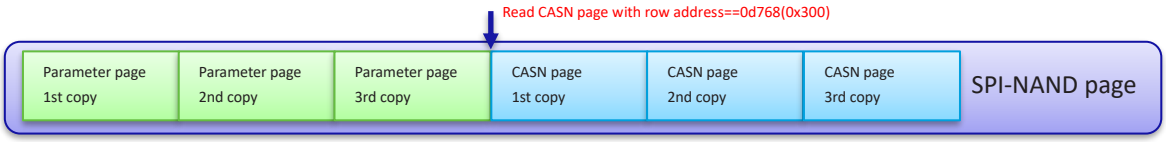
Parameter/CASN Page Row Address	Manufactures
00h	Etron
01h	Dosilicon / ESMT / Fidelix / Foresee / Fudan Micro / GigaSemi(GigaDevice) / Kioxia / Macronix / Micron / Winbond
181h	SkyHigh

Table 1 – CASN Page's row address among manufactures

1.2 Host Driver Programming Guide

There are a few notable things when host driver takes care of CASN page:

1. For readability, CASN utilizes **big endian** encoding.
2. When reading CASN page with OTP read command sets, use correct **page offset** and **row address**. Row address will be 256*3 = 768 (Right after parameter page copies)



3. Necessary check:

CASN Fields	The value must be ("||" means "logical or", "&&" means "logical and")
Bits Per Cell	0x1
Page Size	0x800(0d2048) || 0x1000(0d4096)
OOB Size	0x40(0d64) || 0x60(0d96) || 0x80(0d128) || 0x100(0d256)
Pages Per Block	0x40(0d64) || 0x80(0d128)
Eraseblock Per LUN	0x400(0d1024) || 0x800(0d2048) || 0x1000(0d4096)
Max Bad Blocks Per LUN	0x14(0d20), If "Eraseblock Per LUN" is 0x400(0d1024) 0x28(0d40), If "Eraseblock Per LUN" is 0x800(0d2048) 0x50(0d80), If "Eraseblock Per LUN" is 0x1000(0d4096)
Planes Per LUN	0x1 || 0x2
LUNS Per Target	0x1 || 0x2
Total Targets	0x1 || 0x2
Total OOB Layout	0x0 || 0x1
Advanced ECC Status CMD0/1's Status nBytes	>=0 && <=2

1.3 CASN Page Field Summary

Table 2 describes a whole picture of CASN page.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	Symbol				Version	manufacturer name										
2		model name														
3		bits per cell			page size				(physical) OOB size						pages per block	
4		eraseblock per lun			max bad blocks per lun				logical plaens per lun						luns per target	
5		total targets			ECC strength				ECC step size						flags	reserved
6	**SDR read ability**	SDR 1_1_1 read			SDR 1_1_1 fast read	SDR 1_1_2 read			SDR 1_2_2 read		SDR 1_1_4 read		SDR 1_4_4 read		SDR 1_1_8 read	
		cmd	addr nbytes	dummy nbytes												
7	SDR 1_8_8 read	SDR 1_1_1 cont. read			SDR 1_1_1 cont. fast read	SDR 1_1_2 cont. read			SDR 1_2_2 cont. read		SDR 1_1_4 cont. read		SDR 1_4_4 cont. read		SDR 1_1_8 cont. read	
8	SDR 1_8_8 cont. read	**DDR read ability**			DDR 1_1_1 read	DDR 1_1_1 fast read			DDR 1_1_2 read		DDR 1_2_2 read		DDR 1_1_4 read		DDR 1_4_4 read	
9	DDR 1_1_8 read	DDR 1_8_8 read			DDR 1_1_1 cont. read	DDR 1_1_1 cont. fast read			DDR 1_1_2 cont. read		DDR 1_2_2 cont. read		DDR 1_1_4 cont. read		DDR 1_4_4 cont. read	
10	DDR 1_1_8 cont. read	DDR 1_8_8 cont. read			**SDR write ability**	SDR 1_1_1 write			SDR 1_1_4 write		reserved		reserved		reserved	reser
11	ved	reserved			reserved	**DDR write ability**			reserved		reserved		reserved		reserved	
12	reserved				reserved		**SDR update ability**		SDR 1_1_1 update		SDR 1_1_4 update		reserved		reserved	reser
13	ved	reserved			reserved		reserved	**DDR update ability**			reserved		reserved		reserved	
14	reserved		reserved		reserved		reserved		Total OOB layout	OOB free layout			ECC parity layout			
										OOB free start	OOB free length	BBM length	ECC parity start	ECC parity space	ECC parity (real) length	CMD
15	Advanced ECC status CMD0 (higher bit)										Advanced ECC status CMD1 (lower bit)					
	addr val	addr nbytes	addr buswidth	dummy nbytes	dummy buswidth	status bytes (max=2)	status reg mask		post process operator	post process mask	CMD	addr val	addr nbytes	addr buswidth	dummy nbytes	dummy buswidth
16	Advanced ECC status CMD1 (lower bit)				ECC no error status	ECC uncorrectable status	post porcess operator	post process mask	reserved	reserved	reserved	reserved	reserved	CRC		
	status nbytes (max=2)	status reg mask		post process operator	post process mask											

Table 2 – CASN Page Fields Summary (in bytes)

2 Definitions

2.1 Symbol [0:3]

Symbol must be ASCII code(43h/41h/53h/4Eh) of four characters ('C'/'A'/'S'/'N'). Host driver must check symbol before further parsing CASN page.

2.2 Version [4]

Higher 4 bits represent main version, and lower 4 bits are subversion. For example, version==10h is for version 1.0, version==11h is for version 1.1 and version==23h is for version 2.3. Current CASN version is 1.0. Hereinafter, CASN version 1.0 is abbreviated as **CASN-V1**.

2.3 Manufacturer Name [5:17]

Manufacturers should put their names in this field. 13 bytes should be enough, though.

2.4 Model Name [18:33]

CASN introduces model name field if manufacturers need to maintain lots of SPI-NAND part numbers. Host driver can print out model name so that users can easily know which SPI-NAND flash they mount.

2.5 Bit Per Cell [34:37]

Refer to table 3. As CASN is used for probing SPI-NAND flashes, this field in CASN-V1 must be 00h,00h,00h,01h because nowadays SPI-NAND utilizes small density SLC.

Bit Per Cell	Meaning
00h 00h 00h 01h	SLC
00h 00h 00h 02h	MLC
00h 00h 00h 03h	TLC

Table 3 – Bit Per Cell definition

2.6 Page Size [38:41]

This field indicates Page Size of SPI-NAND, which should be 2048 or 4096 (bytes) in decimal.

2.7 OOB Size [42:45]

This field indicates **"physical"** out-of-boundary/spare area size of SPI-NAND, which should be 64, 96, 128 or 256 (bytes) in decimal.

2.8 Pages Per Block [46:49]

This field indicates page number in each block of SPI-NAND, which should be 64 or 128 in decimal.

2.9 Eraseblock Per LUN [50:53]

This field indicates block numbers in each LUN of SPI-NAND, which should be 1024, 2048 or 4096 in decimal.

2.10 Max Bad Blocks Per LUN [54:57]

This field indicates max bad block number in each LUN of SPI-NAND and it's coupled with "eraseblock per lun" field as table 4 shows:

Eraseblock Per LUN	Max bad blocks per LUN
1024	20
2048	40
4096	80

Table 4 – Copulation of "eraseblock per LUN" & "max bad blocks per LUN" (in decimal)

2.11 Logical Planes Per LUN [58: 61]

This field indicates **"logical"** plane number in each LUN of SPI-NAND, which is important for correct addressing. Some SPI-NAND may have **"physical"** two planes inside but address the whole NAND with **"logical"** one plane. In such case, please fill up this field with 0x1(one plane).

2.12 LUNs Per Target [62:65]

This field indicates LUN number in each die of SPI-NAND.

2.13 Total Targets [66:69]

This field indicates die number of SPI-NAND. However, currently, only Winbond W25M02GV has two targets(dies).

2.14 ECC strength [70:73]

This field indicates that maximum bitflips can be corrected in an ECC encoding/decoding chunk. This is related to SPI-NAND flash on-chip ECC engine and memory cell quality.

2.15 ECC Step Size [74:77]

This field indicates ECC encoding/decoding chunk size. For example, manufacturers usually separate 2KB page into 4 sectors/chunks. In this case, ECC step size will be 2048/4=512.

2.16 Flags [78:79]

Byte-78(79th byte) defines 8 key features of SPI-NAND. Each feature takes one bit. Please refer to table 5 for all bit definitions. Byte-79(80th byte) is reserved for future use. For more details of bit 4/5 of byte 78 flag, see Section 2.32.

Flags	Bit	Key feature
Byte 78	7	ECC algorithm (0: hamming code / 1: BCH)
	6	Is ECC parity readable? (0: no / 1: yes)
	5	Support advanced ECC status or not? (0: no / 1: yes)
	4	Support legacy ECC status or not? (0: no / 1: yes)
	3	Support flash on-chip ECC or not? (0: no / 1: yes)
	2	Support continuous read or not? (0: no / 1: yes)
	1	Does continuous read feature bit exist or not? (0: no / 1: yes)
	0	Does QUAD mode bit exist or not? (0: no / 1: yes)
Byte 79	7-0	Reserved for other flags

Table 5 – Definitions of each bit in CASN-V1 Flags field

2.17 SDR Read Ability [80:81]

Refer to table 6 to see supported single data rate(SDR) read commands of SPI-NAND. “X_Y_Z” means bus width X for SPI command, bus width Y for SPI address and bus width Z for SPI data. For example, 1_8_8 means single bus width for command, octal bus width for address and data. Here we also take continuous read into consideration.

CASN-V1 Byte	Bit	Content
80	15	Support 1_8_8 continuous read or not? (0: no / 1: yes)
	14	Support 1_1_8 continuous read or not? (0: no / 1: yes)
	13	Support 1_4_4 continuous read or not? (0: no / 1: yes)
	12	Support 1_1_4 continuous read or not? (0: no / 1: yes)
	11	Support 1_2_2 continuous read or not? (0: no / 1: yes)
	10	Support 1_1_2 continuous read or not? (0: no / 1: yes)
	9	Support 1_1_1 fast continuous read or not? (0: no / 1: yes)
	8	Support 1_1_1 continuous read or not? (0: no / 1: yes)
81	7	Support 1_8_8 read or not? (0: no / 1: yes)
	6	Support 1_1_8 read or not? (0: no / 1: yes)
	5	Support 1_4_4 read or not? (0: no / 1: yes)
	4	Support 1_1_4 read or not? (0: no / 1: yes)
	3	Support 1_2_2 read or not? (0: no / 1: yes)
	2	Support 1_1_2 read or not? (0: no / 1: yes)
	1	Support 1_1_1 fast read or not? (0: no / 1: yes)
	0	Support 1_1_1 read or not? (0: no / 1: yes)

Table 6 – Definitions of each bit in SDR read ability

Dummy byte's bus width basically follows address bus width. Here's how bus width is defined:

SPI Protocol	Bus width			
	command	address	dummy	data
1_1_1	1	1	1	1
1_1_2	1	1	1	2
1_2_2	1	2	2	2
1_1_4	1	1	1	4
1_4_4	1	4	4	4
1_1_8	1	1	1	8
1_8_8	1	8	8	8

Table 7 – SPI Protocol and its bus width

2.18 SDR 1_1_1 Read [82:83] / SDR 1_1_1 Fast Read [84:85] / SDR 1_1_2 Read [86:87] / SDR 1_2_2 Read [88:89] / SDR 1_1_4 Read [90:91] /SDR 1_4_4 Read [92:93] / SDR 1_1_8 Read [94:95] / SDR 1_8_8 Read [96:97] / SDR 1_1_1 Continuous Read [98:99] / SDR 1_1_1 Fast Continuous Read [100:101] / SDR 1_1_2 Continuous Read [102:103] / SDR 1_2_2 Continuous Read [104:105] / SDR 1_1_4 Continuous Read [106:107] / SDR 1_4_4 Continuous Read [108:109] / SDR 1_1_8 Continuous Read [110:111] / SDR 1_8_8 Continuous Read[112:113]

Each read capability field contain 2 bytes information, including 8-bit command(cmd), 4-bit number of address bytes (addr nbytes) and 4-bit number of dummy bytes (dummy nbytes). The format is described in table 8.

Bits	8-bit	4-bit	4-bit
Definition	cmd	addr nbytes	dummy nbytes
Ranges	0x0~0xff	0x0~0xf	0x0~0xf

Table 8 – Customized SPI-NAND transmission format

That is to say, flash manufacturers can customize their own command (set) and number of address/dummy bytes here. Note that command can be 0x0 here, which is different from Advanced ECC Status CMD0/1 in Section 2.32.

2.19 DDR Read Ability[114:115]

These 2 bytes have similar format like what's mentioned in Section 2.17 but just changes SDR to double data rate (DDR). Only a few SPI-NAND supports this feature nowadays.

2.20 DDR 1_1_1 Read [116:117] / DDR 1_1_1 Fast Read [118:119] / DDR 1_1_2 Read [120:121] / DDR 1_2_2 [122:123] / DDR 1_1_4 Read [124:125] / DDR 1_4_4 Read [126:127] / DDR 1_1_8 Read [128:129] / DDR 1_8_8 Read [130:131] / DDR 1_1_1 Continuous Read [132:133] / DDR 1_1_1 Fast Continuous Read [134:135] / DDR 1_1_2 Continuous Read [136:137] / DDR 1_2_2 Continuous Read [138:139] / DDR 1_1_4 Continuous Read [140:141] / DDR 1_4_4 Continuous Read [142:143] / DDR 1_1_8 Continuous Read [144:145] / DDR 1_8_8 Continuous Read [146:147]

These fields also follow the same format as mentioned in table 8. Flash manufacturers can also customize their own DDR read command (set) and number of address/dummy bytes here.

2.21 SDR Write Ability [148]

Through this byte, host controller can detect which write command, i.e.,“Program Load” command in SPI-NAND, to use. However, currently, flash manufacturers don’t tend to implement x2/dual mode for write operations. Therefore, in CASN-V1, we only define x1/x4 write operations in bit0 and bit1:

CASN-V1 Byte	Bit	Content
148	7	Reserved
	6	Reserved
	5	Reserved
	4	Reserved
	3	Reserved
	2	Reserved
	1	Support 1_1_4 program load or not? (0: no / 1: yes)
	0	Support 1_1_1 program load or not? (0: no / 1: yes)

Table 9 – Definitions of each bit in SDR write ability

2.22 SDR 1_1_1 Write [149:150] / SDR 1_1_4 Write [151:152] / Reserved [153:164]

These fields also follow the same format as mentioned in table 8. Flash manufacturers can also customize their own SDR “Program Load” command (set) and number of address/dummy bytes here. 12 bytes are reserved for future use.

2.23 DDR 1_1_1 Write Ability [165]

This byte has similar format like what's mentioned in Section 2.21 but just changes SDR to DDR. Basically, this is defined for future use.

2.24 Reserved [166:181]

Currently, no SPI-NAND support DDR "Program Load" operations. We reserve 16 bytes for future use.

2.25 SDR Update Ability [182]

This byte has similar format like what's mentioned in Section 2.21 but just changes SDR "write" ability to SDR "update" ability. This usually refers to "Random Program Load" commands in SPI-NAND.

CASN-V1 Byte	Bit	Content
182	7	Reserved
	6	Reserved
	5	Reserved
	4	Reserved
	3	Reserved
	2	Reserved
	1	Support 1_1_4 random program load or not? (0: no / 1: yes)
	0	Support 1_1_1 random program load or not? (0: no / 1: yes)

Table 10 – Definitions of each bit in SDR update ability

2.26 SDR 1_1_1 Update [183:184] / SDR 1_1_4 Update [185:186] / Reserved [187:198]

These fields also follow the same format as mentioned in table 8. Flash manufacturers can also customize their own SDR random program load command (set) and number of address/dummy bytes here. 12 bytes are reserved for future use.

2.27 DDR Update Ability [199]

This byte has similar format like what's mentioned in Section 2.25 but just changes SDR to DDR. Basically, this is defined for future use.

2.28 Reserved [200:215]

Currently, no SPI-NAND support DDR "Random Program Load" operations. We reserve 16 bytes for future use.

2.29 Total OOB Layout[216]

In CASN-V1, there are only two values of "Toal OOB Layout". 0x0 is for "Discrete" and 0x1 is for "Continuous". Out-of-boundary(OOB) area is usually there for storing metadata and flash on-chip's ECC parity bit. However, every flash manufacturer arranges those data in different ways. Take OOB of 64-bytes for example, it may be arranged:

1. In “Discrete” way: OOB free (to use) segments are not adjacent.

8-bytes OOB free	8-bytes ECC parity	8-bytes OOB free	8-bytes ECC parity	8-bytes OOB free	8-bytes ECC parity	8-bytes OOB free	8-bytes ECC parity

2. Or in “Continuous” way: OOB free (to use) segments are adjacent.

8-bytes OOB free	8-bytes OOB free	8-bytes OOB free	8-bytes OOB free	8-bytes ECC parity	8-bytes ECC parity	8-bytes ECC parity	8-bytes ECC parity

Note that some SPI-NAND may arrange OOB in different ways when flash’s internal ECC is on and off. This field is determined under the circumstance that the flash’s internal ECC is on. When flash’s internal ECC is off, OOB layout is determined by host controller’s ECC engine.

2.30 OOB Free Layout [217:219]

2.30.1 OOB Free Start [217]

This byte indicates start offset of OOB free area where users can store data. This is usually 0x0. However, some legacy SPI-NAND design may occupy some previous bytes of OOB area and makes OOB free start offset shifting to 0x1 or later. That’s why CASN-V1 adds this field to help host driver to locate start offset to store user metadata.

2.30.2 OOB Free Length [218]

Besides ECC parity bit segments, OOB free are also sliced into a few segments. For all pages in flash, “OOB Free Length” field indicates the length of each OOB free segments. However, for the 1st OOB free segments of the 1st page of a certain block, it may contain bad block mark(BBM). Host driver should handle this and take away the length of BBM from “OOB Free Length” to calculate the length of 1st OOB free segment.

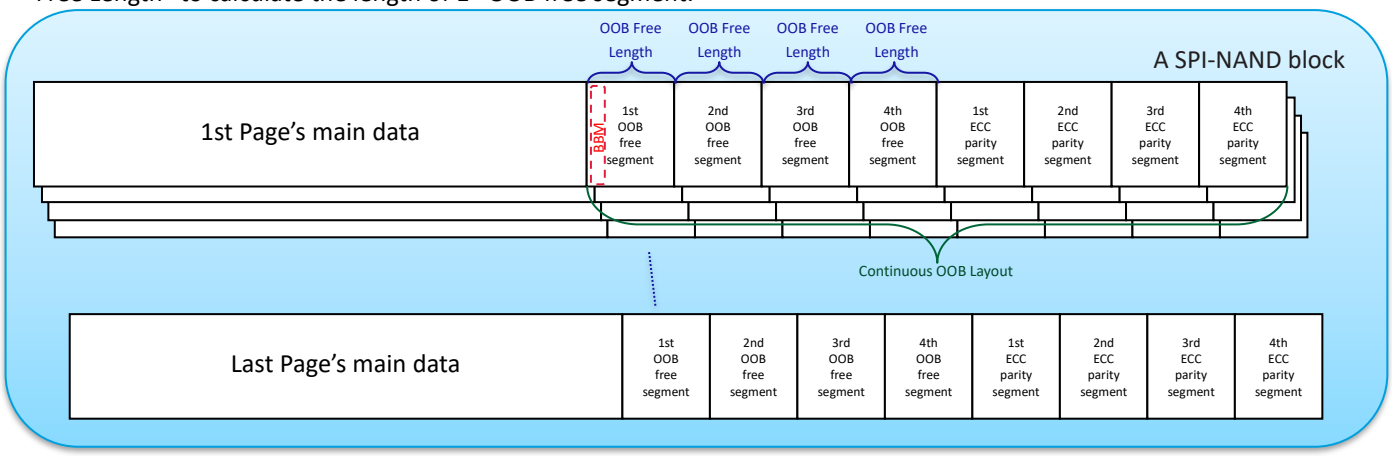


Image 4 – OOB Layout Demonstration in one SPI-NAND block

2.30.3 BBM Length [219]

Bad block mark(BBM) is used to indicate whether a block is bad or not. In Linux kernel, BBM usually takes 2 bytes. However, we still add a field for flash manufacturers to customize their BBM length since not everyone is using Linux kernel.

2.31 ECC Parity Layout [220:222]

2.31.1 ECC Parity Start [220]

ECC parity start will be start offset of 1st ECC parity segment. However, this field is related to “Total OOB Layout”. In image 5 and 6, they are both 64 bytes OOB area. You can see different ECC parity start value with different OOB layout.

2.31.2 ECC Parity Space [221]

This field shows maximum space(bytes) that flash on-chip ECC engine can use to store its ECC parity data. Refer to Image 5 and 6 to see details.

2.31.3 ECC Parity Real Length [222]

This field shows real ECC parity length generated by flash on-chip ECC engine. In Image 5 and 6, you can see that real ECC parity bit may not occupy the whole ECC parity space.

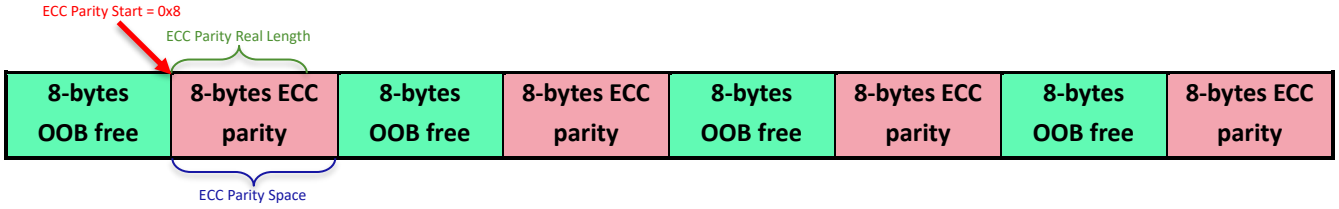


Image 5 – ECC Parity Layout in 64 bytes Discrete OOB layout

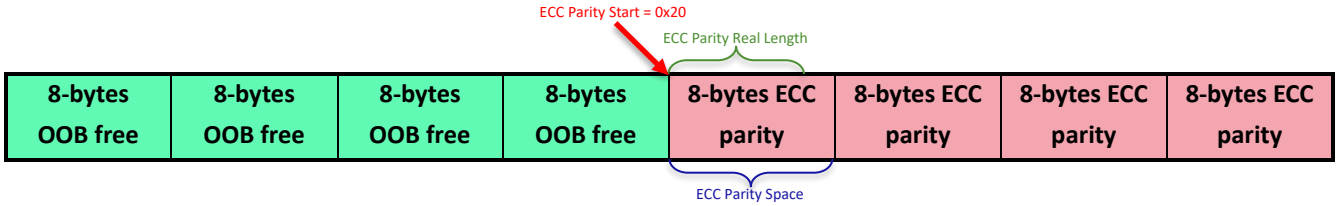


Image 6 – ECC Parity Layout in 64 bytes Continuous OOB layout

Note that Image 5 and 6 are examples for 64 bytes OOB layout. If your physical OOB size is not 64 bytes, reference the following table 11 to find out ECC parity start.

Page Size	Physical OOB Size	Total OOB Layout	ECC Step Size	Number of OOB sections	OOB Free Length	ECC Parity Start
2048	64	Discrete	512	4	8	8
2048	64	Continuous	512	4	8	32
2048	96	Discrete	512	4	16	16
2048	96	Continuous	512	4	16	64
2048	128	Discrete	512	4	16	16
2048	128	Continuous	512	4	16	64
2048	128	Discrete	512	4	18	18
2048	128	Continuous	512	4	18	72
4096	256	Discrete	512	8	16	16
4096	256	Continuous	512	8	16	64

Table 11 – ECC Parity Start reference table (in decimal)

2.32 Advanced ECC Status CMD0 [223:233] / Advanced ECC Status CMD1 [234:244]

In CASN-V1, we design a mechanism to transform SPI-NAND on-chip ECC status into unified format, which will give host controller precise bitflip information. Most Legacy SPI-NAND has 2-bit ECC status register to represent 3 states: “No bitflip”, “Correctable bitflips” and “Uncorrectable bitflips”. Here’s how they are defined in Linux Kernel at “include/linux/mtd/spinand.h”:

```
#define STATUS_ECC_NO_BITFLIPS   (0 << 4)
#define STATUS_ECC_HAS_BITFLIPS  (1 << 4)
#define STATUS_ECC_UNCOR_ERROR   (2 << 4)
```

We call this **“Legacy” ECC status** in CASN-V1. Note that some manufacturers may adopt one more state, 0x3, to represent vendor specific content. Take Etron EM73C044VCF-H and Winbond W25N01GV for example:

ECCS1, ECCS0	ECC Status	This bit provides ECC status as follows: 00b = No bit errors were detected 01b = bit error was detected and corrected 10b = bit error was detected and not corrected 11b = bit error was detected and corrected, error bit number = ECC max which is according to extended register. ECCS is set to 00b either following a RESET, or at the beginning of the READ. It is then updated after the device completes a valid operation. After power-on RESET, ECC status is set to reflect the contents of block 0, page 0.

Image 7 – Etron EM73C044VCF-H ECC status table

7.3.2 Cumulative ECC Status (ECC-1, ECC-0) – *Status Only*

ECC function is used in NAND flash memory to correct limited memory errors during read operations. The ECC Status Bits (ECC-1, ECC-0) should be checked after the completion of a Read operation to verify the data integrity. The ECC Status bits values are don't care if ECC-E=0. These bits will be cleared to 0 after a power cycle or a RESET command or a Page Data Read command.

ECC Status		Descriptions
ECC-1	ECC-0	
0	0	The entire data output is provided without requiring any ECC correction.
0	1	The entire data output experienced a 1 bit correction event after reading either single or multiple pages.
1	0	The entire data output experienced a 2 bit error event in a single page that cannot be corrected[2] in the Continuous Read Mode, an additional command can be used to read out the Page Address (PA) that contains the error.
1	1	The entire data output experienced a 2 bit error event in multiple pages. In the Continuous Read Mode, the additional command can only provide the last Page Address (PA) that contain the 2 bit error. PAs for other pages with the 2 bit error is not available. The data read is not suitable for use[2,3].

Image 8 – Winbond W25N01GV ECC status table

Before the end of life of SPI-NAND flash, it will experience three stages: No bitflip → more and more bitflips → Too many bitflips and they are uncorrectable. This is state 0x0 → 0x1 → 0x2 according what we mentioned above. During this period, if another state 0x3 will take place, please use Advanced ECC Transform Mechanism in Section 3 instead. Take Etron EM73C044VCF-H for example, its SPI-NAND lifecycle will be: No bitflip(0x0) → more and more bitflips(0x1) → Reach ECC engine’s max ability(0x3) → Too many bitflips and they are uncorrectable (0x2). That’s why Etron EM73C044VCF-H only supports advanced ECC status in CASN-V1. On the contrary, Winbond W25N01GV’s lifecycle is: No bitflip(0x0) → more and more bitflips(0x1) → Too many bitflips and they are uncorrectable (0x2) in single page → Too many bitflips and they are uncorrectable (0x3) in multiple pages. In this device, 0x3 ECC state will not take place before uncorrectable bitflips happen. So it’s okay to use legacy ECC status.

However, in this way, SPI-NAND on-chip ECC engine won’t give correct number of bitflip taking place. Therefore, some manufacturers like GigaSemi(GigaDevice), Macronix and Winbond, add more ECC status bit or another (8-bit) ECC status register to represent real bitflip numbers. We call this **“Advanced”(or “Enhanced”) ECC status** in CASN-V1.

For example, in GigaSemi(GigaDevice) GD5F1GM7 datasheet, it defines another two ECCSE1 and ECCSE0 bit:

Table 12-3. ECC Error Bits Descriptions

ECCS1	ECCS0	ECCSE1	ECCSE0	Description
0	0	x	x	No bit errors were detected during the previous read algorithm
0	1	0	0	Bit errors(≤4) were detected and corrected
0	1	0	1	Bit errors (=5) were detected and corrected.
0	1	1	0	Bit errors (=6) were detected and corrected.
0	1	1	1	Bit errors (=7) were detected and corrected.
1	1	x	x	Bit errors (=8) were detected and corrected.
1	0	x	x	Bit errors greater than ECC capability(8 bits) and not corrected

In Macronix MX35LF1GE4AB, it defines 7Ch command to read a brand-new ECCSR (internal ECC status register):

Table 6-2. The Definition of Internal ECC Status

ECCSR[3:0]	ECC Status
0000	No bit error
0001	1-bit error corrected
0010	2-bit error corrected
0011	3-bit error corrected
0100	4-bit error corrected
1111	Uncorrectable

The Winbond W25N01KV datasheet specifies additional bits for provision of enhanced ECC statistical data:

7.4 Extended internal ECC feature registers

Address	Bit							
	S7	S6	S5	S4	S3	S2	S1	S0
10h	I	BFD2	BFD1	BFD0	I	I	I	I
20h	I	I	I	I	BFS3	BFS2	BFS1	BFS0
30h	I	MBF2	MBF1	MBF0	I	MFS2	MFS1	MFS0
40h	I	BFR6	BFR5	BFR4	I	BFR2	BFR1	BFR0
50h	I	BFR14	BFR13	BFR12	I	BFR10	BFR9	BFR8

* I = Reserved Bit

Figure 9. Extended Internal ECC feature registers

Unfortunately, all these registers have different format. That's why we need a mechanism to transform them into unified expression. The mechanism utilizes Advanced ECC Status CMD0 and CMD1 in companion with "ECC No Error Status", "ECC Uncorrectable Status", "Post Process Operator" and "Post Process Mask". See more details in Section 3.

For convenience, we abbreviate "Advanced ECC Status CMD(0/1)" as **ADVECC-CMD(0/1)** in later sections.

2.32.1 CMD [223][234]

Flash manufacturers can customize their own command (except 0x0) in this field to read special advanced(enhanced) ECC status registers. For example, it will be 7Ch for this field in Macronix MX35LF1GE4AB's CASN page.

If this value is 0x0, it means that we don't need to trigger this command.

2.32.2 Address Value [224][235]

Specify address value of ADVECC-CMD in this field. This indicates the address of special enhanced ECC status registers.

2.32.3 Address nBytes [225][236]

Number of address bytes for ADVECC-CMD. If this value is 0x1, it means 1 byte for address.

2.32.4 Address Buswidth [226][237]

Of course, bus width of address should be also specified. This is usually 0x1.

2.32.5 Dummy nBytes [227][238]

It might take some dummy bytes for some SPI-NAND operating at high speed clock to read out status registers. Flash manufacturer can add dummy bytes for ADVECC-CMD if needed.

2.32.6 Dummy Buswidth [228][239]

And yes, we need to specify bus width for dummy bytes if they exist.

2.32.7 Status nBytes [229][240]

This field ranges from 0x0~0x2. Advanced ECC Status CMD0/1 can read out 2 bytes data at most in CASN-V1. This can express 2^16=65536 number of bitflips, which is pretty enough for SLC SPI-NAND.

2.32.8 Status Register Mask [230:231][241:242]

After reading advanced(enhanced) ECC status registers, “Status Register Mask” is used to derive bits we want. For example, if this value is 0x00F0, it means that we want to take out bit[7:4] from advanced(enhanced) ECC status registers.

2.32.9 Pre Process Operator [232][243]

In CASN-V1, we define 5 operators for pre process and post process (Section 2.35), which is described in table 12.

2.32.10 Pre Process Mask [233][244]

This field defines the operand for pre process operator. The relations are described in table 12.

Pre/Post Process Operators	Operators	Pre/Post Process Mask	Meaning
0x0	None	0x1	Do nothing
0x1	Bitwise AND	0x1	& 0x1
0x2	Addition	0x1	+ 0x1
0x3	Subtraction	0x1	- 0x1
0x4	Multiplication	0x1	* 0x1

Table 12 – Pre Process of ADVECC-CMD

2.33 ECC No Error Status [245]

We'll combine the values after pre-processing from ADVECC-CMD0 and ADVECC-CMD1 reads, which results in virtual Final Advanced ECC Status. Flash manufacturers can define their own "ECC No Error Status" to indicate that no bitflip happens. In Image 7 of Section 3, if the virtual Final Advanced ECC Status is equal to "ECC No Error Status" defined, host driver should end transform mechanism and show that no bitflip takes place.

2.34 ECC Uncorrectable Status [246]

This works the same way as "ECC No Error Status" in Section 2.33. Flash manufacturer **must** define this "ECC Uncorrectable Status" to notify host controller once SPI-NAND on-chip ECC engine can't handle bitflips.

2.35 Post Process Operator [247]

In CASN-V1, we define the same 5 operators as mentioned in Section 2.32.9.

2.36 Post Process Mask [248]

This field defines the operand for post process operators. The relations also follow table 12.

2.37 Reserved [249:253]

Reserve 5 bytes for future use.

2.38 CRC [254:255]

In CASN-V1, Cycling Redundancy Check(CRC) also follows big-endian. This is used to verify that the contents of CASN page were transferred correctly to the host. CASN CRC utilizes the same CRC16 mechanism as ONFI 1.0 specification, but the initial CRC value in the shift register will be changed from "ON" to "CA". The CRC shall be calculated using the following 16-bit generator polynomial: G(X) = X16+X15+X2+1

Here's a C code implementation:

```
#include <stdio.h>

#define CASN_CRC_BASE  (0x4341)
#define CASN_V1_LEN    (256)
#define CASN_CRC_OFS   (CASN_V1_LEN-2)

static __uint16_t casn_crc16(__uint16_t crc, __uint8_t const *p, __uint32_t len)
{
        __uint32_t i;

        while (len--) {
                crc ^= *p++ << 8;
                for (i = 0; i < 8; i++) {
                        crc = (crc << 1) ^ ((crc & 0x8000) ? 0x8005 : 0);
                }
        }

        return crc;
}

int main(int argc, char**argv)
{
        if(argc < 2) {
                printf("Please provide a filename as an argument.\n");
```

```
                return 1;
        }
        FILE* file = fopen(argv[1], "r");
        if(!file) {
                perror("Failed to open file");
                return 1;
        }

          uint8 t num[CASN CRC OFS];
        size_t len = sizeof(num)/sizeof(__uint8_t);
        for(int i = 0; i < len; ++i) {
                if(fscanf(file, "%hhx", &num[i]) != 1) {
                        printf("Failed to read byte %d\n", i);
                        fclose(file);
                        return 1;
                }
        }

        __uint16_t crc_compute;
        __uint8_t *p = (__uint8_t *)&crc_compute;
        crc_compute = casn_crc16(CASN_CRC_BASE, num, CASN_CRC_OFS);
        printf("CRC computed: 0x%04x\n", crc_compute);
        printf("Arrange CRC in big endian:\n");
        printf(" - byte254: 0x%02x\n - byte255: 0x%02x\n", *p, *p++);
        fclose(file);

        return 0;
}
```

3 Advanced ECC Transform Mechanism

According to Section 2.32, we know that we can issue ADVECC-CMD0 and ADVECC-CMD1 to read out “two” advanced (enhanced) ECC status registers. If there is only one advanced(enhanced) ECC status register in SPI-NAND, we only need to issue ADVECC-CMD1. On the contrary, fields in Advanced ECC Status CMD0 contain all 0x0 then.

The concept of the mechanism is to compose a virtual final status register (refer to Final Advanced ECC Status in Image 7), in which the value exactly represents the number of bitflips. However, this is just concept. Let’s break down the transform mechanism implementation into 4 detailed steps:

- Step 1: Use ADVECC-CMD0 or ADVECC-CMD1 to read advanced(enhanced) ECC status registers.
- Step 2: After reading out registers, mask them with “Status Register Mask”(Section 2.32.8) first. And then pre-process it with “Pre Process Operator” and “Pre Process Mask”. This will compose a virtual Final Advanced ECC Status.
- Step 3: If the value in virtual Final Advanced ECC Status meets “ECC No Error Status” defined in Section 2.33 or “ECC Uncorrectable Status” defined in Section 2.34, just return the status.
- Step 4: If step 3 doesn’t happen, keep post-processing the virtual Final Advanced ECC Status value with operators and mask defined in Section 2.35 and 2.36. We’ll then return this value, which represents the number of bitflips. If the value exceeds flash on-chip ECC engine’s max correction ability, i.e., ECC strength in Section 2.14 , host driver should report ECC strength to host system instead.

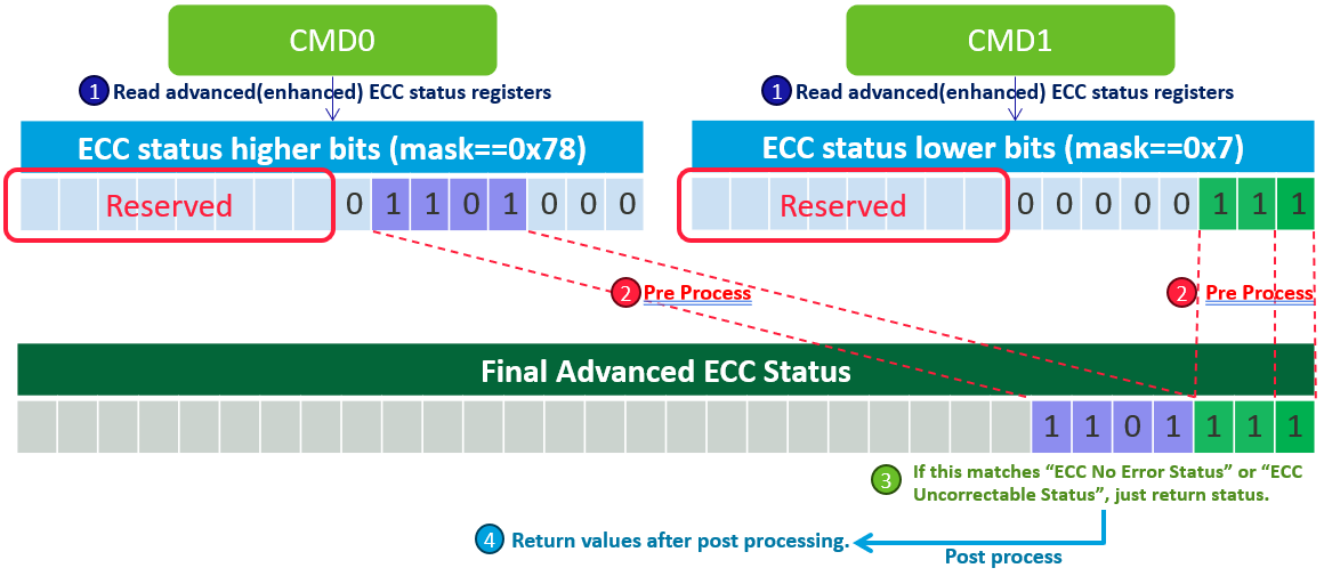


Image 9 – Structure of advanced ECC transform mechanism

Take GigaSemi(GigaDevice) GD5F1GQ5UExxG for example, it has ECCS1/ECCS0/ECCSE1/ECCSE0 ECC status registers:

Table 12-1. Features Settings

Register	Addr.	7	6	5	4	3	2	1	0
Protection	A0H	BRWD	Reserved	BP2	BP1	BP0	INV	CMP	Reserved
Feature	B0H	OTP_PRT	OTP_EN	Reserved	ECC_EN	BPL	Reserved	Reserved	QE
Status	C0H	Reserved	Reserved	ECCS1	ECCS0	P_FAIL	E_FAIL	WEL	OIP
Feature	D0H	Reserved	DS_IO[1]	DS_IO[0]	Reserved	Reserved	Reserved	Reserved	Reserved
Status	F0H	Reserved	Reserved	ECCSE1	ECCSE0	BPS	Reserved	Reserved	Reserved

Image 10 – ECC status register location of GigaSemi(GigaDevice) GD5F1GQ5UExxG

These 4 bits from 2 registers compose the following status:

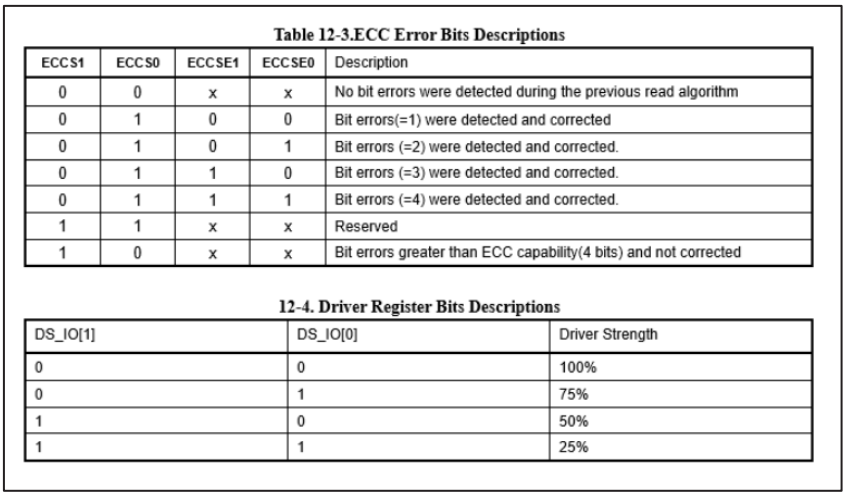


Table 12-3.ECC Error Bits Descriptions

ECCS1	ECCS0	ECCSE1	ECCSE0	Description
0	0	x	x	No bit errors were detected during the previous read algorithm
0	1	0	0	Bit errors(=1) were detected and corrected
0	1	0	1	Bit errors (=2) were detected and corrected.
0	1	1	0	Bit errors (=3) were detected and corrected.
0	1	1	1	Bit errors (=4) were detected and corrected.
1	1	x	x	Reserved
1	0	x	x	Bit errors greater than ECC capability(4 bits) and not corrected

12-4. Driver Register Bits Descriptions

DS_IO[1]	DS_IO[0]	Driver Strength
0	0	100%
0	1	75%
1	0	50%
1	1	25%

Image 11 – ECC status table of GigaSemi(GigaDevice) GD5F1GQ5UExxG (On real chip, “x” will “0”)

- Step 1: Use ADVECC-CMD0 to read ECCS1 and ECCS0 with address 0xC0 and ADVECC-CMD1 to read ECCSE1 and ECCSE0 with address 0xF0. We’ll get 2 bytes data like this:

Register	Addr.	7	6	5	4	3	2	1	0
Status	C0H	Reserved	Reserved	ECCS1	ECCS0	P_FAIL	E_FAIL	WEL	OIP

Register	Addr.	7	6	5	4	3	2	1	0
Status	F0H	Reserved	Reserved	ECCSE1	ECCSE0	BPS	Reserved	Reserved	Reserved

- Step 2: Mask ECCS1 and ECCS0 with “Status Register Mask” 0x0030 (Section 2.32.8, CASN page byte 230-231). Also mask ECCSE1 and ECCSE0 with “Status Register Mask” 0x0030 (Section 2.32.8, CASN page byte 241-242). Both ADVECC-CMD0’s and ADVECC-CMD1’s “Pre Process Operator” and “Pre Process Mask” are 0x0, so we don’t need any pre-processing. Then we may get 6 kinds of state according to image 11 (get rid of reserved state):

ECCS1	ECCS0	ECCSE1	ECCSE0	Virtual Final Advanced ECC Status (in hexadecimal)
0	0	0	0	0x0
0	1	0	0	0x4
0	1	0	1	0x5
0	1	1	0	0x6
0	1	1	1	0x7
1	0	0	0	0x8

- Step 3: If the “virtual Final Advanced ECC Status” is 0x0, it hits “ECC No Error Status” and host driver just reports that no bitflip happens. If the “virtual Final Advanced ECC Status” is 0x8, it hits “ECC Uncorrectable Status” and host driver will know that the page/block is broken. As a result, there are 4 states left:

ECCS1	ECCS0	ECCSE1	ECCSE0	Virtual Final Advanced ECC Status (in hexadecimal)
0	1	0	0	0x4
0	1	0	1	0x5
0	1	1	0	0x6
0	1	1	1	0x7

- Step 4: GigaSemi(GigaDevice) GD5F1GQ5UExxG's "Post Process Operator" is 0x3 and "Post Process Mask" is 0x3, which means "minus three". And we'll get the "Virtual Final Advanced ECC Status" after post-processing:

Virtual Final Advanced ECC Status (in hexadecimal)	**After post-processing** (host driver will report this as the number of bitflips)
0x4	0x1
0x5	0x2
0x6	0x3
0x7	0x4

4 CASN-V1 Compatible List

Part number	Size	Support Legacy ECC?	Support Advanced ECC?	Datasheet
ESMT				
F50L1G41LB	1Gb	Yes	No	
F50L2G41KA	2Gb	No (3 ECC status bits)	Yes	
Etron				
EM73C044VCF-H	1Gb	No (Reach 0x3 state before uncorrectable bitflips)	Yes	
EM73D044VCO-H	2Gb	No (Reach 0x3 state before uncorrectable bitflips)	Yes	
EM73E044VCE-H	4Gb	No (Reach 0x3 state before uncorrectable bitflips)	Yes	
EM73F044VCA-H	8Gb	No (Reach 0x3 state before uncorrectable bitflips)	Yes	
GigaSemi (GigaDevice)				
GD5F1GM7UE (Mainstream)	1Gb	No (0x3 state means “reserved”)	Yes	
GD5F1GQ5UEYIG	1Gb	Yes	Yes	
GD5F2GM7UE (Mainstream)	2Gb	No (0x3 state means “reserved”)	Yes	
GD5F2GQ5UEYIG	2Gb	Yes	Yes	
GD5F4GM8UE (Mainstream)	4Gb	No (0x3 state means “reserved”)	Yes	
GD5F4GQ6UEYIG	4Gb	Yes	Yes	
Macronix (MXIC)				
MX35LF1GE4ABZ4IG	1Gb	Yes	Yes	Link
Winbond				
W25N01GV	1Gb	Yes	No	
W25N01KV	1Gb	No (Reach 0x3 state before uncorrectable bitflips)	Yes	
W25N02KV	2Gb	No (Reach 0x3 state before uncorrectable bitflips)	Yes	
W25N04KV	4Gb	No (Reach 0x3 state before uncorrectable bitflips)	Yes	

5 Special Thanks

Thanks for the concerted efforts of all cooperated partners and flash manufactures who help promoting and testing. We look forward to more flash manufacturers considering the addition of CASN page.

- MediaTek: Steven.Liu / Ada.Huang
- ESMT: Venessa.Hsu / YC.Kuo / Kelly.Cheng / Ken.Weng
- Etron: Felix.Lin / Minhou.Lee / Tommy.Tang/ Irene.Liao
- GigaSemi(GigaDevice): Dannis.Chang / Ian.lin / Congzhen.Liu / Baiquan.Li
- MXIC: Henry.Ho / Psyche(ulkuo)
- Winbond: Steam.Lin / Vivian.Cheng