

VR13 / VR13.HC / VR14 multiphase VR porting guide

Texas Instruments

July 2022

Ann Lien

Contents

- FW programming solutions
- Porting concept from system
- Porting steps

FW programming solution

- Off line programming by 3rd party
- Off line programming by customer's factory
- On line programming by customer's factory
- On line programming with BIOS by customer's system

FW programming solution

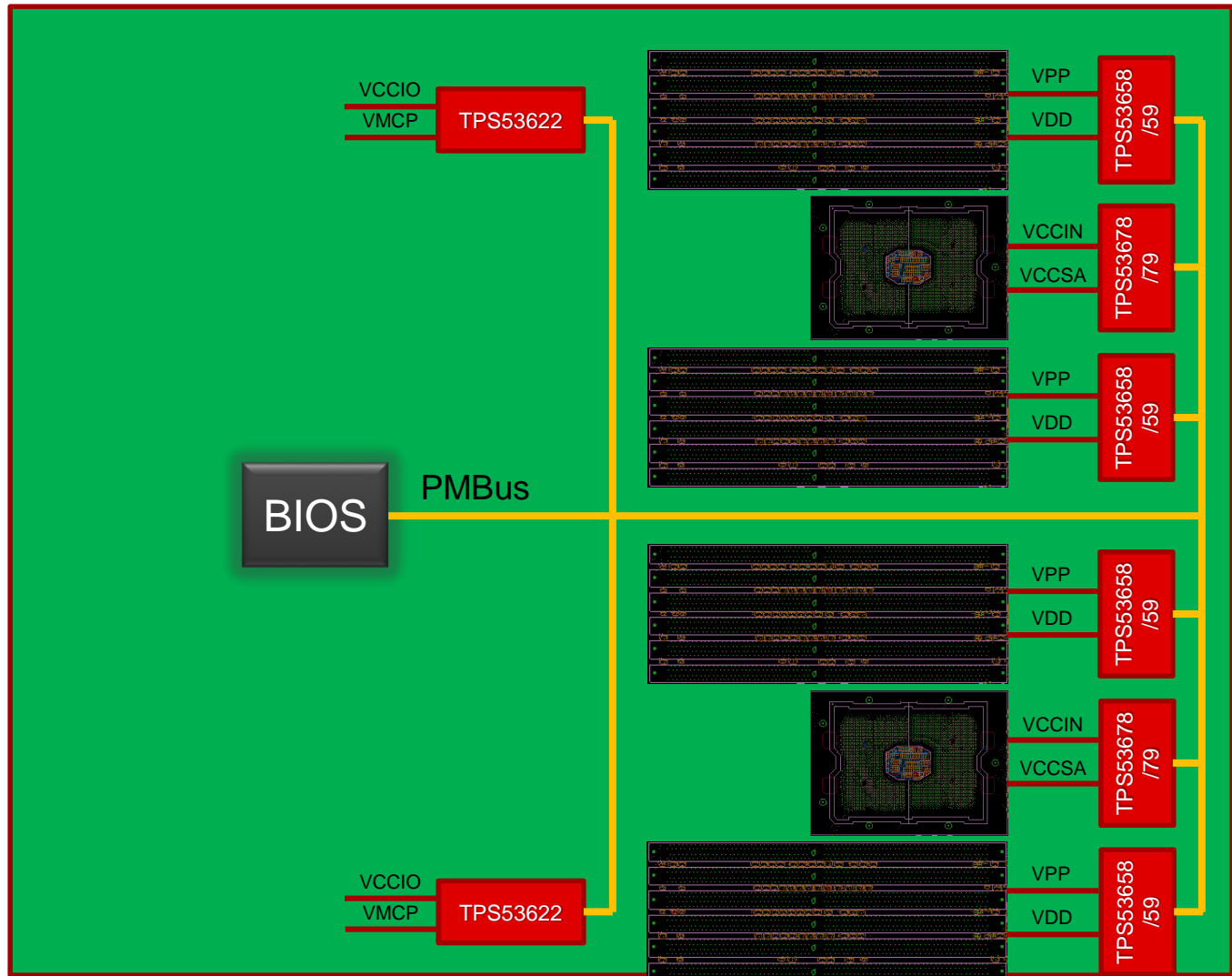
- Advantages and Disadvantages

	Offline by 3 rd party	Offline by customer factory	Online by customer factory	Online by customer system BIOS
Advantages	<ol style="list-style-type: none"> 1. Easy for assembly. 	<ol style="list-style-type: none"> 1. Less part no. 	<ol style="list-style-type: none"> 1. Less part no. 	<ol style="list-style-type: none"> 1. Less part no. 2. Easy for assembly. 3. Saving programming cost.
Disadvantages	<ol style="list-style-type: none"> 1. Higher IC cost. 2. More part no. 	<ol style="list-style-type: none"> 1. Higher production cost/effort. 	<ol style="list-style-type: none"> 1. Higher production cost/effort. 	<ol style="list-style-type: none"> 1. No. (only BIOS FW RD effort)

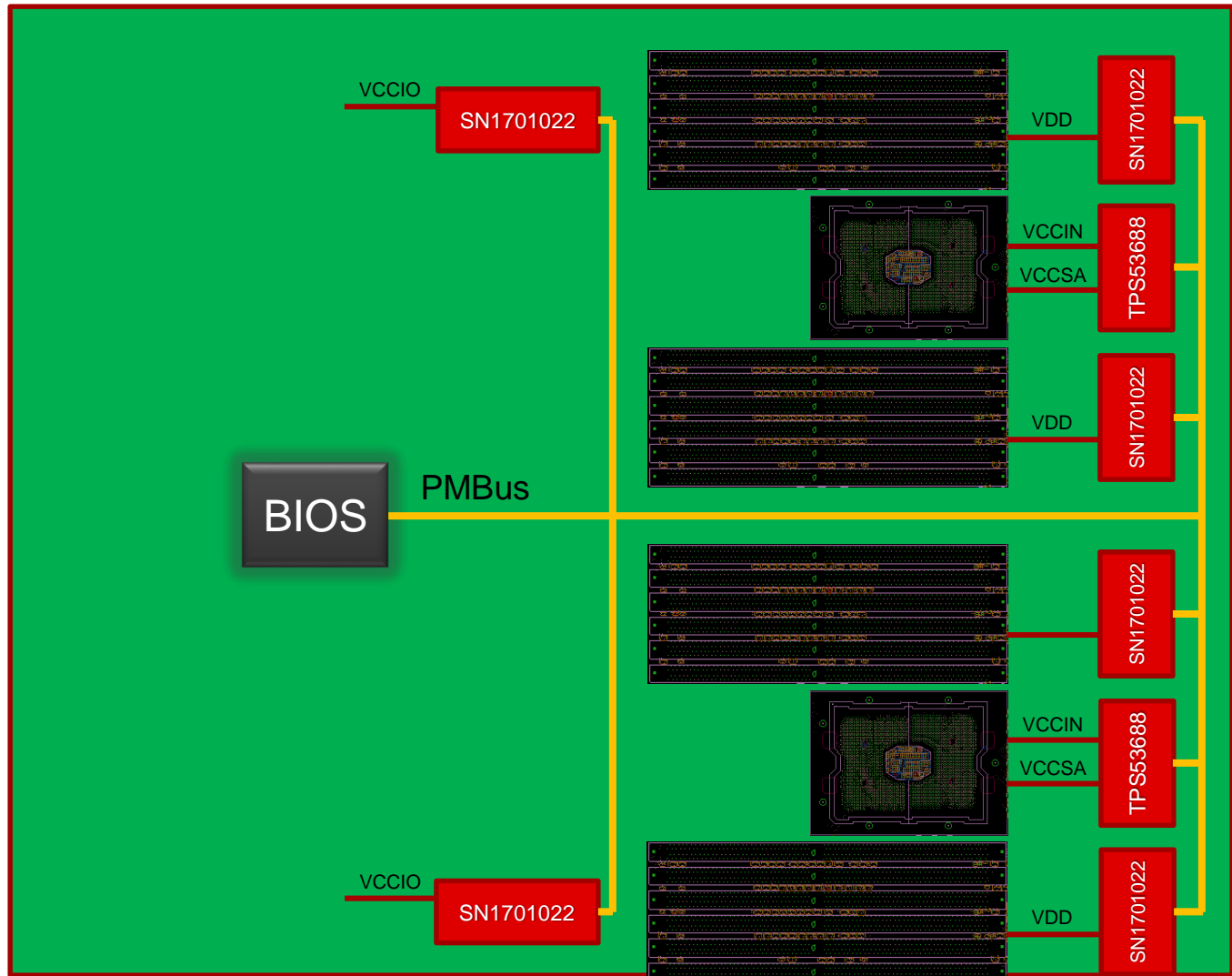
Porting concept from system

- In general, customer often online or offline to program the parameters to NVM of VR IC.
- Customer got higher cost for production if they choose online programming. If they choose offline solution, the procurement staff need to buy more items (because IC with different parameters) that increase stock cost.
- The porting concept is saving parameters in BIOS and load to the RAM of VR IC through PMBus when motherboard startup every time.
- The benefits are
 - Saving production cost.
 - Minimized items to different project.
 - RD still keep the optimized parameters to every project.

VR13 Porting concept from system

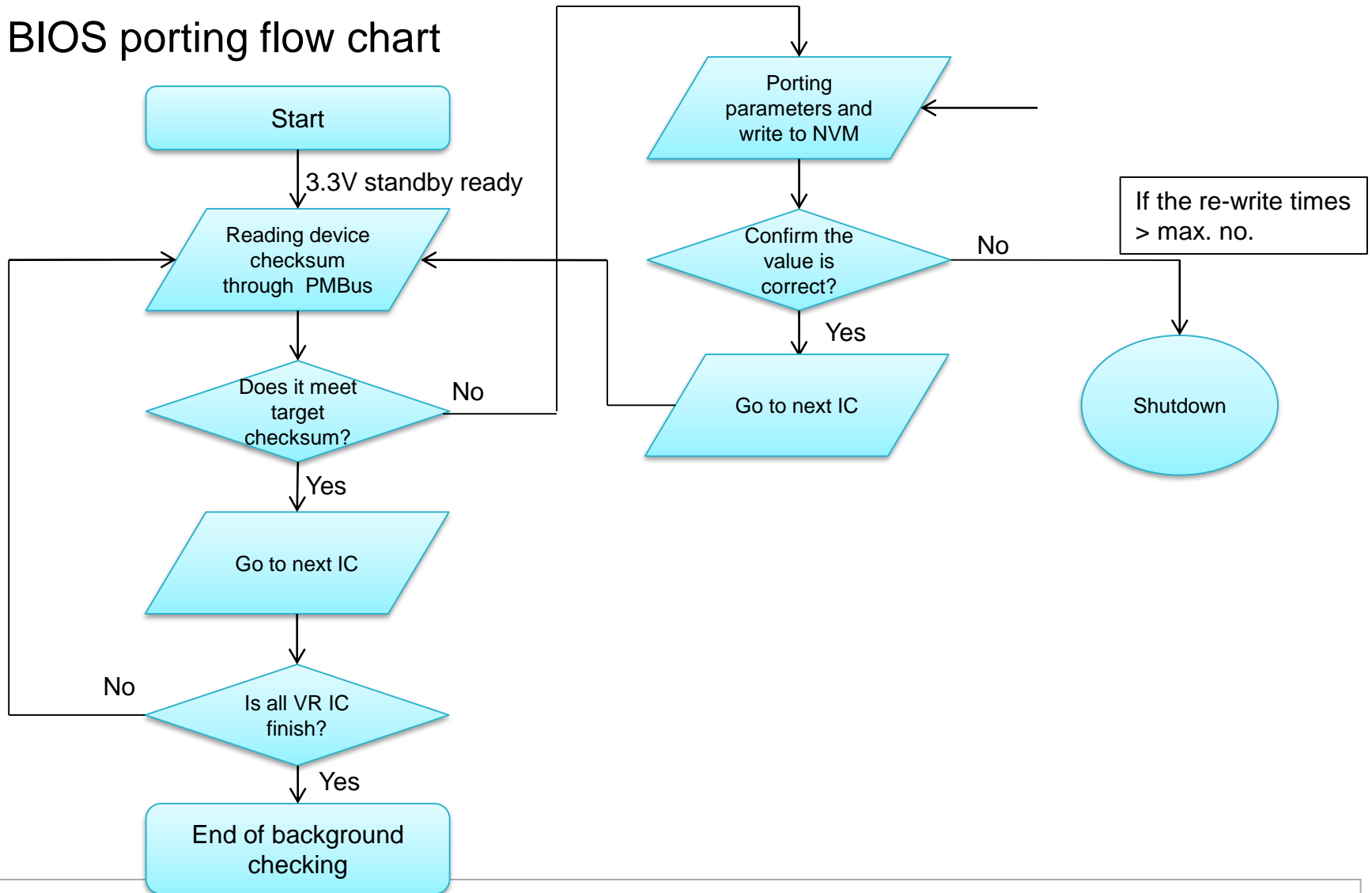


VR13.HC Porting concept from system



Porting steps

- BIOS porting flow chart



VR13(SN1701021/2) Porting SOP

- **Step 1**

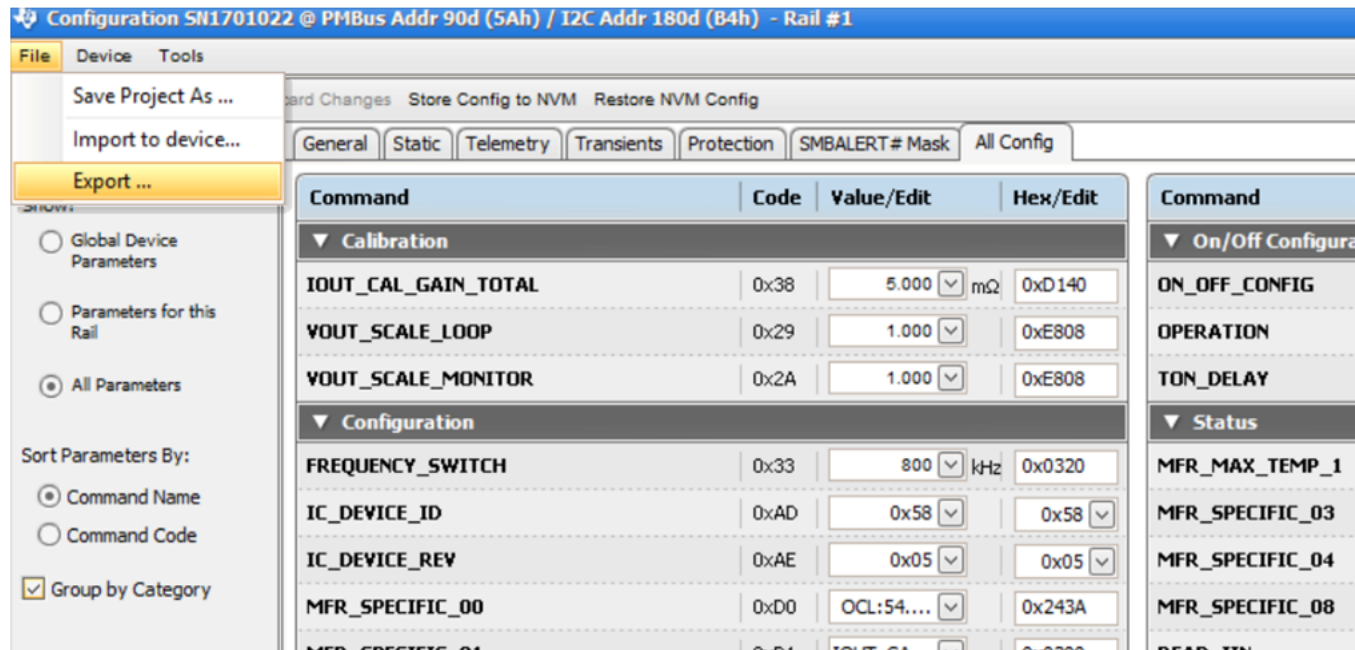
- Go to All Config
- Write register MFR_ID [0x99] → User Data 00 (Example : E1)
- Write register MFR_REVISION [0x9B] → User Data 01 (Example : 53)
- Write to Hardware
- Store to NVM

Manufacturer Info			
CAPABILITY	0x19	0xD0	0xD0
MFR_DATE	0x9D	Oct 2016	0x100A
MFR_ID	0x99	E1	0x4531
MFR_IOUT_MAX	0xA6	0 A	0x0000
MFR_MODEL	0x9A		0x0000
MFR_PIN_MAX	0xA3	120 W	0x003C
MFR_REVISION	0x9B	53	0x3533
MFR_SERIAL	0x9E	7A64D718	0x7A...
MFR_VOUT_MAX	0xA5	0.000 V	0x0000
MFR_VOUT_MIN	0xA4	0.000 V	0x0000
PMBUS_REVISION	0x98	0x33	0x33

VR13(SN1701021/2) Porting SOP

- **Step 2**

- Go into configure device
- File → Export



VR13(SN1701021/2) Porting SOP

- **Step 3**

- Select **PMBus Programmer Script**
- **Don't change** any check box or button
- Click **Export PMBus Programmer Script**

Device Export - SN1701022 @ PMBus Address 90d

Export Multiple Formats: Text File, Project File, **PMBus Programmer Script**

Description
This will save a script detailing the writes necessary to write your current configuration to a device. Writes are done in terms of standard SMBus commands (WriteByte, WriteWord, and WriteBlock). This can be easily translated to other environments.

Options

Configuration Validation

- A validation step is performed after a device reset to read back the configuration and verify it matches what was programmed. The script is oriented at third party programmers who can support resetting the device after the configuration is written to non-volatile memory.
- Do not validate configuration

Write Validation

- Validate command writes
The script will read back commands after they are written to verify writes. You should definitely use this option if your microcontroller does not have the capability to check for NACK on write.
- Do not validate command writes

Programming options

- Turn rails off before programming
Some devices require rails to be off before programming. If selected, all rails on the target device will be turned off first by writing ON_OFF_CONFIG command (Code=02h) with value = 00h (OPERATION Only), followed by writing OPERATION command (Code=01h) with value = 00h (Immediate Off) to all rails
- Add IC_DEVICE_ID (0xAD) to script
If selected, IC_DEVICE_ID (0xAD) is read and added to the script. Your program can verify if target device's IC_DEVICE_ID (0xAD) matches the one in the script before continue with programming.
- Add MFR_SERIAL validation to script
If selected, after programming, and configuration is stored to flash, your program will read back the MFR_SERIAL from the device and verify the value against the MFR_SERIAL stored in the script.

Other options

- Including Device PMBus address
If select, script will include device address in the second column
- Store Default Timing: ms
The delay time after the STORE_DEFAULT_ALL command is sent before continue to the next step
- Add PEC byte
Bytes order in PMBusWord commands:
 - <low-byte><high-byte>[PEC byte]
 - <high-byte><low-byte>Add PEC byte option is not available for this selection

Output Destination

Output Folder: C:\Users\A0223156\Desktop

Filename: {PN} {DV} Address {DA} {EF}.{EXT}

Preview: SN1701022 5.0 Address 90 PMBus Programmer Script.csv

Log

11:11:11.733: Stopped background polling

VR13(SN1701021/2) Porting SOP

- **Step 4**

- Collect USER DATA 0~12, VOUT_MAX[Rail#1] and VOUT_MAX[Rail#2] value

Comment	Write VOUT_MAX [Rail #1] 1.520 V		
WriteByte	0x00	0x0041	
ReadByte	0x00	0x00	
WriteWord	0x24	0xFF00FF	
ReadWord	0x24	0xFF00	

Comment	Write VOUT_MAX [Rail #2] 3.040 V		
WriteByte	0x00	0x0146	
ReadByte	0x00	0x01	
WriteWord	0x24	0xFF00FF	
ReadWord	0x24	0xFF00	

Comment	Write USER_DATA_00 [Rail #1] 1184030050C3		
BlockWrite	0xB0	0x061184030050C3AB	
BlockRead	0xB0	0x061184030050C3	
Comment	Write USER_DATA_01 [Rail #1] 000000000040		
BlockWrite	0xB1	0x06000000000040A1	
BlockRead	0xB1	0x06000000000040	
Comment	Write USER_DATA_02 [Rail #1] 890200000000		
BlockWrite	0xB2	0x068902000000001D	
BlockRead	0xB2	0x06890200000000	
Comment	Write USER_DATA_03 [Rail #1] 0064100A16C0		
BlockWrite	0xB3	0x060064100A16C0E4	
		.	
		.	
		.	
Comment	Write USER_DATA_10 [Rail #1] 02602880C18D		
BlockWrite	0xBA	0x0602602880C18D12	
BlockRead	0xBA	0x0602602880C18D	
Comment	Write USER_DATA_11 [Rail #1] 00622000A021		
BlockWrite	0xBB	0x0600622000A02181	
BlockRead	0xBB	0x0600622000A021	
Comment	Write USER_DATA_12 [Rail #1] 8C500200F801		
BlockWrite	0xBC	0x068C500200F801CB	
BlockRead	0xBC	0x068C500200F801	

VR13(SN1701022) Porting SOP

- **Step 5**

- Check [Checksum] in MFR_SERIAL (0x9E)

General	Static	Telemetry	Transients	Protection	SMBALERT# Mask	All Config
▼ Manufacturer Info						
CAPABILITY	0x19	0xD0	0xD0			
MFR_DATE	0x9D	Oct 2016	0x100A			
MFR_ID	0x99	E1	0x4531			
MFR_IOUT_MAX	0xA6	0 A	0x0000			
MFR_MODEL	0x9A	53	0x3533			
MFR_PIN_MAX	0xA3	120 W	0x003C			
MFR_REVISION	0x9B		0x0064			
MFR_SERIAL	0x9E	4937C34B	0x49...			
MFR_VOUT_MAX	0xA5	0.000 V	0x0000			
MFR_VOUT_MIN	0xA4	0.000 V	0x0000			
PMBUS_REVISION	0x98	0x33	0x33			

VR13 Porting file example

- **Step 6**

- Fill in previous value into porting file.



VR13 example

Step	Action	Register command	Register address	Data	Command type	Description
1	Write	PAGE	00h	00	R/W	Switch to page 0.
2	Write	USER_DATA_00	B0h	1184030050C3 (example)	R/W, NVM	Write data to USER_DATA_00.
3	Write	USER_DATA_01	B1h	000000000040 (example)	R/W, NVM	Write data to USER_DATA_01.
4	Write	USER_DATA_02	B2h	890200000000 (example)	R/W, NVM	Write data to USER_DATA_02.
5	Write	USER_DATA_03	B3h	0064100A16C0 (example)	R/W, NVM	Write data to USER_DATA_03.
6	Write	USER_DATA_04	B4h	4424C777C777 (example)	R/W, NVM	Write data to USER_DATA_04.
7	Write	USER_DATA_05	B5h	8332C5BFCF3C (example)	R/W, NVM	Write data to USER_DATA_05.
8	Write	USER_DATA_06	B6h	E51B0F160080 (example)	R/W, NVM	Write data to USER_DATA_06.
9	Write	USER_DATA_07	B7h	80FF091033FF (example)	R/W, NVM	Write data to USER_DATA_07.
10	Write	USER_DATA_08	B8h	00020000008D (example)	R/W, NVM	Write data to USER_DATA_08.
11	Write	USER_DATA_09	B9h	0001608080AE (example)	R/W, NVM	Write data to USER_DATA_09.
12	Write	USER_DATA_10	BAh	02602880C18D (example)	R/W, NVM	Write data to USER_DATA_10.
13	Write	USER_DATA_11	BBh	00622000A021 (example)	R/W, NVM	Write data to USER_DATA_11.
14	Write	USER_DATA_12	BCh	8C500200F801 (example)	R/W, NVM	Write data to USER_DATA_12.
15	Write	VOUT_MAX	24h	FF00 (example)	R/W, NVM	Write data to VOUT_MAX of page 0.
16	Write	PAGE	00h	01	R/W	Switch to page 1.
17	Write	VOUT_MAX	24h	FF00 (example)	R/W, NVM	Write data to VOUT_MAX of page 1.
18	Write	STORE_DEFAULT_ALL	11h		W	Stores all current storable register setting into NVM as new defaults.
19	Read	MFR_SERIAL	9Eh	4937C34B (example)	R	If the checksum matched (4937C34B), the NVM programming is correct.

VR13.HC/VR14 (TPS53688/9) Porting SOP

- **Step 1**

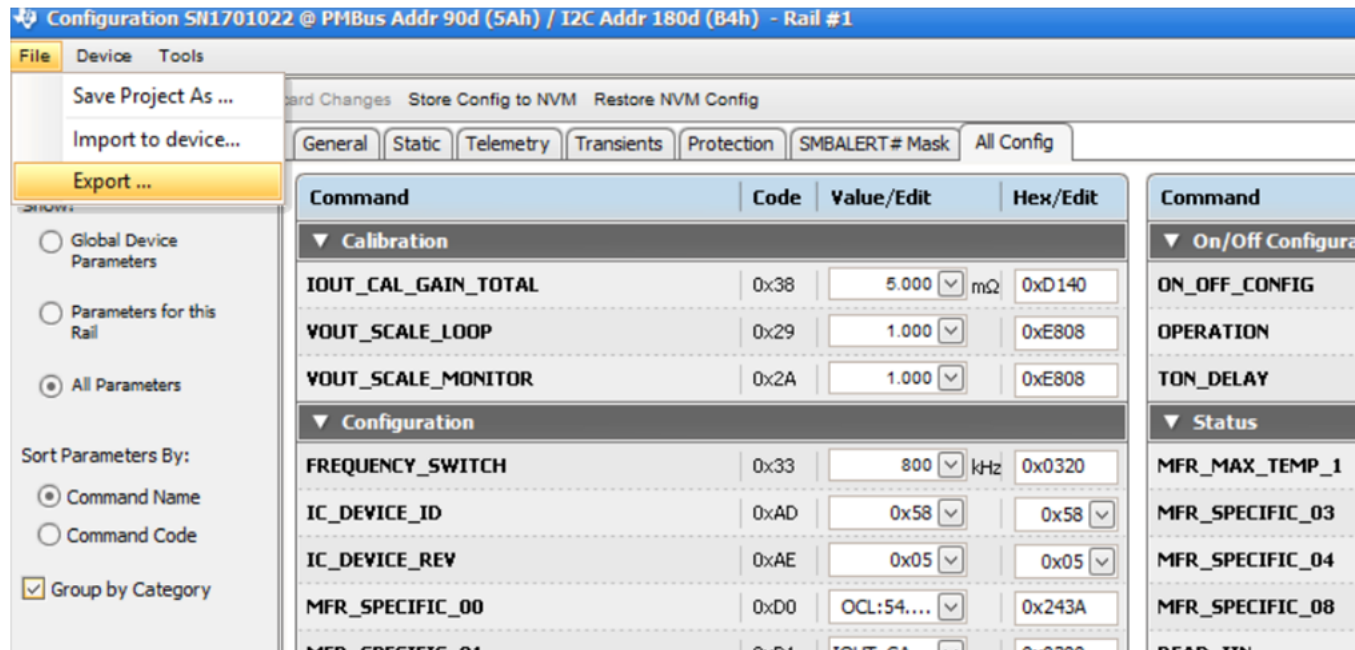
- Go to All Config
- Write register MFR_ID [0x99] → User Data 00 (Example : 0E1)
- Write register MFR_REVISION [0x9B] → User Data 01 (Example : 053)
- **Please add one more character – 0 in front of User Data 00 and 01**
- Write to Hardware
- Store to NVM

System Configuration	Rail Configuration	Protection	Calibration	SVID Configuration	SMBAlert Mask	NVM Programming	All Config
VOUT_UV_WARN_LIMIT	0x43	0.312	v	0x005A			
▼ Manufacturer Info							
CAPABILITY	0x19	0xD0		0xD0			
MFR_DATE	0x9D			0x00...			
MFR_ID	0x99	0E1		0x30...			
MFR_MODEL	0x9A			0x00...			
MFR_REVISION	0x9B	053		0x30...			
PMBUS_REVISION	0x98	0x33		0x33			

VR13.HC/VR14 (TPS53688/9) Porting SOP

- **Step 2**

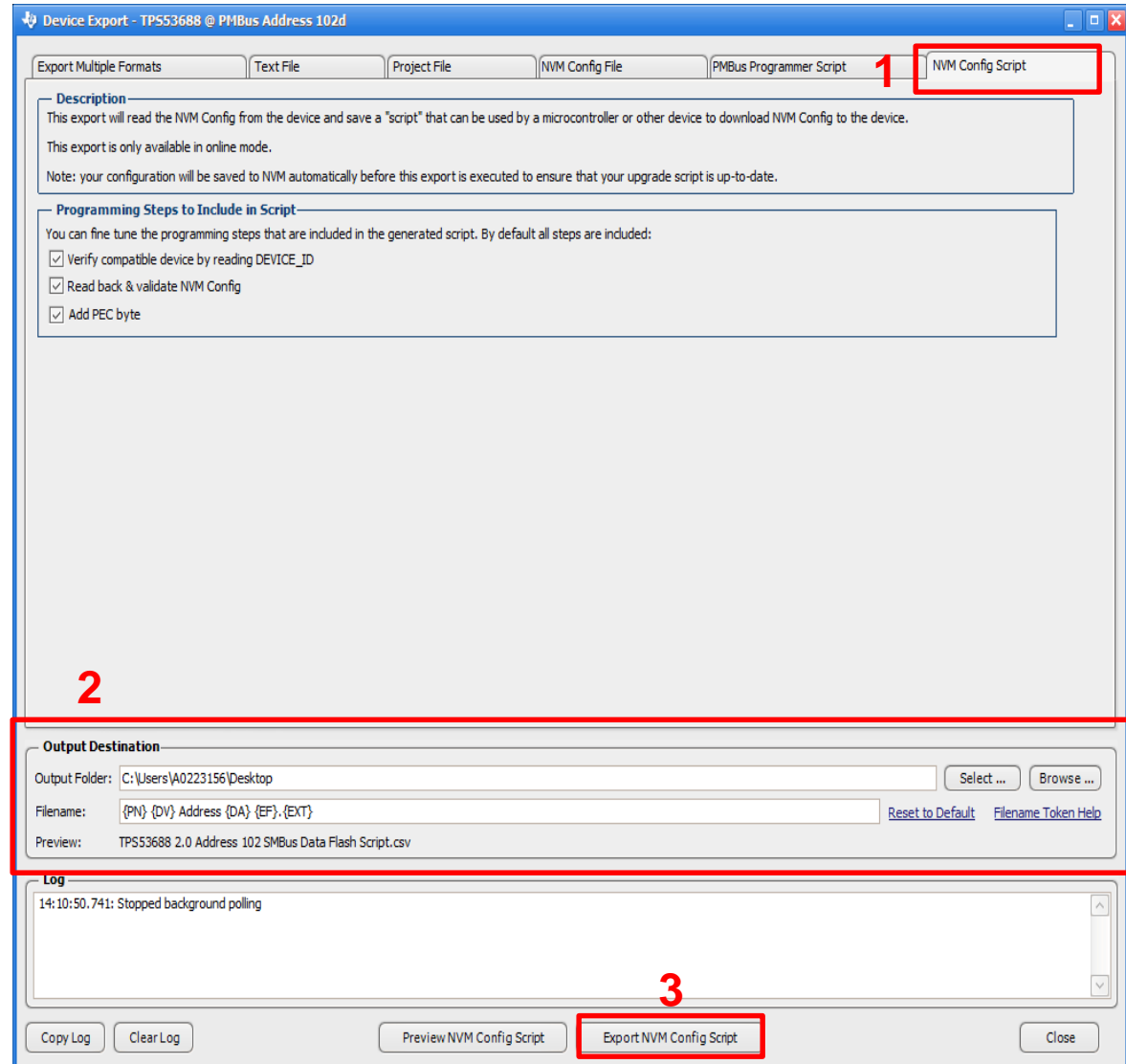
- Go into configure device
- File → Export



VR13.HC/VR14 (TPS53688/9) Porting SOP

- **Step 3**

- Select **NVM Config Script**
- **Don't change** any check box or button
- Select **Output Destination**
- Click **Export NVM Config Script**



VR13.HC/VR14 (TPS53688/9) Porting SOP

- Step 4

- Collect Register F5 and F6 index 0~8 value(as in red square)

Comment Write USER_NV_M_INDEX [MFR_37]	0x00				
WriteByte 0x66	0xF5	0x0022			
Comment Write USER_NV_M [MFR_38], index = 0, value = 0x	FFFFFFFFFFFFFFFFD8F51717000000004000024277900830000008A00FF00				
BlockWrite 0x66	0xF6	0x20FFFFFFFFFFFFFFFFD8F51717000000004000024277900830000008A00FF0008			
Comment Write USER_NV_M [MFR_38], index = 1, value = 0x	C957400C00000000CADC746040406060202020606060612060C52512C52512				
BlockWrite 0x66	0xF6	0x20C957400C00000000CADC746040406060202020606060612060C52512C52512F2			
Comment Write USER_NV_M [MFR_38], index = 2, value = 0x	CF5519089014514F5250535150010101010202010104045382C8000002550000				
BlockWrite 0x66	0xF6	0x20CF5519089014514F5250535150010101010202010104045382C80000025500009E			
Comment Write USER_NV_M [MFR_38], index = 3, value = 0x	0000814007A0388100D0E10102D91382B3B262B0D1033FF0F0731440F0FFA80				
BlockWrite 0x66	0xF6	0x200000814007A0388100D0E10102D91382B3B262B0D1033FF0F0731440F0FFA80B3			
Comment Write USER_NV_M [MFR_38], index = 4, value = 0x	8284818385869000000000C040082222200000444444444444F0000088470002				
BlockWrite 0x66	0xF6	0x208284818385869000000000C040082222200000444444444444F000008847000242			
Comment Write USER_NV_M [MFR_38], index = 5, value = 0x	AE00000EDAFE4964010E140064FF50120A0A000062FFFF0000DA0000030E0101				
BlockWrite 0x66	0xF6	0x20AE00000EDAFE4964010E140064FF50120A0A000062FFFF0000DA0000030E010168			
Comment Write USER_NV_M [MFR_38], index = 6, value = 0x	010101010101010141012E2A038D000010000000000060000000000002A0B14				
BlockWrite 0x66	0xF6	0x200101010101010141012E2A038D000010000000000060000000000002A0B148B			
Comment Write USER_NV_M [MFR_38], index = 7, value = 0x	1000800EE220C80BB001B00001B008FFFFFFFFF87C487C403000000304531				
BlockWrite 0x66	0xF6	0x201000800EE220C80BB001B00001B008FFFFFFFFF87C487C40300000030453146			
Comment Write USER_NV_M [MFR_38], index = 8, value = 0x	30353300				
BlockWrite 0x66	0xF6	0x2030353300E9			

VR13.HC/VR14 (TPS53688/9) Porting SOP

- **Step 5**

- Check [**Checksum**] in NVM_CHECKSUM (0xF0 for TPS53688, 0xF4 for TPS53689)

System Configuration	Rail Configuration	Protection	Calibration	SVID Configuration	SMBAlert Mask	NVM Programming	All Config
VOUT_OFFSET_VID [MFR 28]	0xEC	0.0000	mV	0xB000			
VOUT_SCALE_LOOP	0x29	1.000		0xE808			
▼ Configuration							
FREQUENCY_SWITCH	0x33	800	kHz	0x0320			
IC_DEVICE_ID	0xAD	0x54495...		0x54...			
IC_DEVICE_REV	0xAE	0x0002		0x0002			
MFR_SPECIFIC_WRITE_PROTECT	0xFB	WP_ALL:...		0x0000			
MISC_OPTIONS [MFR 29]	0xED	REQPEC:...		0xD...			
NVM_CHECKSUM [MFR 32]	0xF0	NVM_CH...		0xF5D8			
PIN_DETECT_OVERRIDE [MFR 30]	0xEE	0x1A		0x1A			
POWER_MODE	0x34	0x00		0x00			
SLAVE_ADDRESS [MFR 31]	0xEF	0x66		0x66			
SMBALERT_MASK_CML	0x1B	00000000		0x00			
SMBALERT_MASK_EXTENDED	0xCF	0x00000...		0x00...			
SMBALERT_MASK_INPUT	0x1B	00000100		0x04			

VR13.HC/VR14 (TPS53688/9) Porting SOP

- **Step 6-1**

- Fill in previous value into porting file(TPS53688).



VR13.HC example

Step	Action	Register command	Register address	Data
1	Write byte	USER_NVM_INDEX	F5h	0x00 (example)
2	Write block	USER_NVM_EXECUTE (INDEX 0)	F6h	FFFFFFFFFFFFFFFFD8F51717000000004000024277900830000008A00FF00 (example)
3	Write block	USER_NVM_EXECUTE (INDEX 1)	F6h	C957400C00000000CACD746040406060202020606060612060C52512C52512 (example)
4	Write block	USER_NVM_EXECUTE (INDEX 2)	F6h	CF5519089014514F5250535150010101010202010104045382C8000002550000 (example)
5	Write block	USER_NVM_EXECUTE (INDEX 3)	F6h	0000814007A03881000D0E10102D91382B3B262B0D1033FF0F0731440F0FFA80 (example)
6	Write block	USER_NVM_EXECUTE (INDEX 4)	F6h	8284818385869000000000C0400822222000004444444444444444F0000088470002 (example)
7	Write block	USER_NVM_EXECUTE (INDEX 5)	F6h	AE00000EDAFE4964010E140064FF50120A0A000062FFFF0000DA0000030E0101 (example)
8	Write block	USER_NVM_EXECUTE (INDEX 6)	F6h	010101010101010141012E2A038D00001000000000060000000000002A0B14 (example)
9	Write block	USER_NVM_EXECUTE (INDEX 7)	F6h	10000800EE220C80BB001B00001B008FFFFFFFFFFFF87C487C403000000304531 (example)
10	Write block	USER_NVM_EXECUTE (INDEX 8)	F6h	30353300 (example)
11	Read	NVM_CHECKSUM	F0h	F5D8 (example)

VR13.HC/VR14 (TPS53688/9) Porting SOP

- **Step 6-2**

- Fill in previous value into porting file(TPS53689).



VR14 example

Action	Register command	Register address	Data	Command type	Description
Write byte	USER_NVM_INDEX	F5h	0x00 (example)	R/W, NVM	Set USER_NVM_INDEX at 0
Write block	USER_NVM_EXECUTE (INDEX 0)	F6h	FFFFFFFFFFFFFFFFF8F51717000000004000024277900830000008A00FF00 (example)	R/W, NVM	Write data to USER_NVM_EXECUTE INDEX0
Write block	USER_NVM_EXECUTE (INDEX 1)	F6h	C957400C00000000CADC746040406060202020606060612060C52512C52512 (example)	R/W, NVM	Write data to USER_NVM_EXECUTE INDEX1
Write block	USER_NVM_EXECUTE (INDEX 2)	F6h	CF5519089014514F5250535150010101010202010104045382C8000002550000 (example)	R/W, NVM	Write data to USER_NVM_EXECUTE INDEX2
Write block	USER_NVM_EXECUTE (INDEX 3)	F6h	0000814007A03881000D0E10102D91382B3B262B0D1033FF0F0731440F0FFA80 (example)	R/W, NVM	Write data to USER_NVM_EXECUTE INDEX3
Write block	USER_NVM_EXECUTE (INDEX 4)	F6h	8284818385869000000000C040082222200004444444444F0000088470002 (example)	R/W, NVM	Write data to USER_NVM_EXECUTE INDEX4
Write block	USER_NVM_EXECUTE (INDEX 5)	F6h	AE00000EDAFE4964010E140064FF50120A0A000062FFFF0000DA0000030E0101 (example)	R/W, NVM	Write data to USER_NVM_EXECUTE INDEX5
Write block	USER_NVM_EXECUTE (INDEX 6)	F6h	010101010101010141012E2A038D0000100000000006000000000002A0B14 (example)	R/W, NVM	Write data to USER_NVM_EXECUTE INDEX6
Write block	USER_NVM_EXECUTE (INDEX 7)	F6h	10000800EE220C80BB001B00001B008FFFFFFFFF87C487C403000000304531 (example)	R/W, NVM	Write data to USER_NVM_EXECUTE INDEX7
Write block	USER_NVM_EXECUTE (INDEX 8)	F6h	30353300 (example)	R/W, NVM	Write data to USER_NVM_EXECUTE INDEX8
Read	NVM_CHECKSUM	F4h	F5D8 (example)	R	If the checksum matched (F5D8), the NVM programming is correct.