NXP Semiconductors
Application Note

Document Number: AN5330
Rev 0, 06/2017

Migration Guide for S12ZVM Devices

by: NXP Semiconductors

Contents

1 Introduction

The S12ZVM family gives the smallest, most efficient and scalable BLDC/ PMPSM motor control solution for industrial and automotive applications.

It integrates a sophisticated MCU together with a 12 V to 5 V voltage regulator, LIN physical layer and Gate Driver Unit (GDU) in order to control six power MOSFETs for automotive and industrial applications, such as HVAC blowers, fuel, or water pumps.

The S12ZVM family introduces three packages and five scales of internal memory. Two communication driver options of LIN/CAN are available. Users can select the best option based on the target application with optimized costs.

Usual practice is to start developing an application using the development kit available at nxp.com and then to migrate the application to a custom solution. Another case is to migrate the existing application to a different package in order to reduce PCB size and/or system costs. This application guides user through the migration process from one S12ZVM to another S12ZVM microcontroller. Migrating between the two devices may require hardware and/or software changes. As an example, the document describes the changes required when migrating the PMSM single shunt sensorless software from S12ZVML128 to S12ZVML31. However, the differences between other devices are described as well.

1.1 Part numbering and mask set information

The table below lists the S12ZVM devices. The shaded part ID numbers are not production mask sets.

Table 1. S12ZVM numbering, mask set and feature set key differences

Device	Mask set number	Part ID	Option	Memory Flash/RAM	Package	ADC chan.	Op-amps	TIM chan.
MC9S12ZVMC256	0N00R	0x00180000	CAN	256/32	80 pin	8+8	2	4 + 2
MC9S12ZVMC256	1N00R	0x00180100	CAN	256/32	80 pin	8+8	2	4 + 2

Table continues on the next page...

Table 1. S12ZVM numbering, mask set and feature set key differences (continued)

Device	Mask set number	Part ID	Option	Memory Flash/RAM	Package	ADC chan.	Op-amps	TIM chan.
MC9S12ZVML12	N06E	0x00170000	LIN	128/8	64 pin	4+5	2	4
MC9S12ZVMC12	N06E	0x00170001	CAN-VREG	128/8	64 pin	4+5	2	4
MC9S12ZVML12	0N95G	0x00172000	LIN	128/8	64 pin	4+5	2	4
MC9S12ZVMC12	0N95G	0x00172001	CAN	128/8	64 pin	4+5	2	4
MC9S12ZVML12	1N95G	0x00172100	LIN	128/8	64 pin	4+5	2	4
MC9S12ZVML64	1N95G	0x00172100	LIN	64/4	64 pin	4+5	2	4
MC9S12ZVML32	1N95G	0x00172100	LIN	32/4	64 pin	4+5	2	4
MC9S12ZVMC12	1N95G	0x00172101	CAN-VREG	128/8	64 pin	4+5	2	4
MC9S12ZVMC64	1N95G	0x00172101	CAN-VREG	64/4	64 pin	4+5	2	4
MC9S12ZVML12	2N95G	0x00172200	LIN	128/8	64 pin	4+5	2	4
MC9S12ZVML64	2N95G	0x00172200	LIN	64/4	64 pin	4+5	2	4
MC9S12ZVML32	2N95G	0x00172200	LIN	32/4	64 pin	4+5	2	4
MC9S12ZVMC12	2N95G	0x00172201	CAN-VREG	128/8	64 pin	4+5	2	4
MC9S12ZVMC64	2N95G	0x00172201	CAN-VREG	64/4	64 pin	4+5	2	4
MC9S12ZVML12	3N95G	0x00172300	LIN	128/8	64 pin	4+5	2	4
MC9S12ZVML64	3N95G	0x00172300	LIN	64/4	64 pin	4+5	2	4
MC9S12ZVML32	3N95G	0x00172300	LIN	32/4	64 pin	4+5	2	4
MC9S12ZVMC12	3N95G	0x00172301	CAN-VREG	128/8	64 pin	4+5	2	4
MC9S12ZVMC64	3N95G	0x00172301	CAN-VREG	64/4	64 pin	4+5	2	4
MC9S12ZVML31	0N14N	0x00150000	LIN	32/4	64/48 pin	4+5/1+3	2/1	4/3
MC9S12ZVM32	0N14N	0x00150000	HV Phy	32/4	64/48 pin	4+5/1+3	2/1	4/3
MC9S12ZVM16	0N14N	0x00150000	HV Phy	16/2	64/48 pin	4+5/1+3	2/1	4/3
MC9S12ZVML31	1N14N	0x00150100	LIN	32/4	64/48 pin	4+5/1+3	2/1	4/3
MC9S12ZVM32	1N14N	0x00150100	HV Phy	32/4	64/48 pin	4+5/1+3	2/1	4/3
MC9S12ZVM16	1N14N	0x00150100	HV Phy	16/2	64/48 pin	4+5/1+3	2/1	4/3

1.2 Package differences

When migrating to a different package of S12ZVM, the application should consider changes of the peripherals which may or may not routed to physical pins. Table 2. Pin differences between packages of S12ZVM devices on page 3 shows the module pins and corresponding package pins.

Table 2. Pin differences between packages of S12ZVM devices

Module	Module pin	Pin function	Pin # in 80 LQFP	Pin # in 64C LQFP	Pin # in 64L LQFP	Pin # in 48 LQFP
BDC	BKGD	BKGD / MODC	80	2	2	2
RGTE	RESET	RESET	65	54	54	40
	TEST	TEST	68	18	18	14
GDU	LS0	LS0	43	32	32	23
	LG0	LG0	44	33	33	24
	VLS0	VLS0	45	34	34	-
	VBS0	VBS0	46	35	35	25
	HG0	HG0	47	36	36	26
	HS0	HS0	48	37	37	27
	HS2	HS2	49	38	38	28
	HG2	HG2	50	39	39	29
	VBS2	VBS2	51	40	40	30
	VLS2	VLS2	52	41	41	31
	LG2	LG2	53	42	42	32
	LS2	LS2	54	43	43	33
	LS1	LS1	55	44	44	34
	LG1	LG1	56	45	45	35
	VLS1	VLS1	57	46	46	-
	VBS1	VBS1	58	47	47	36
	HG1	HG1	59	48	48	37
	HS1	HS1	60	49	49	38
	HD	HD	7	10	10	6
	VCP	VCP	6	11	11	7
	BST	BST	5	12	12	8
	VSSB	VSSB	4	13	13	9
	CP	CP	3	14	14	10
	VLS_OUT	VLS_OUT	2	15	15	11
	LD0	LD0	16	-	-	-
	LD1	LD1	17	-	-	-
	LD2	LD2	18	-	-	-
VREG	VSUP	VSUP	1	16	16	12

Table continues on the next page...

Table 2. Pin differences between packages of S12ZVM devices (continued)

Module	Module pin	Pin function	Pin # in 80 LQFP	Pin # in 64C LQFP	Pin # in 64L LQFP	Pin # in 48 LQFP
	VDDX2	VDDX2	-	17	17	13
	VSS2	VSS2	-	19	19	15
	VDD	VDD	75	20	20	16
	VDDA	VDH0_1 / VRH1_1	28	30	30	21
	VSSA	VRL0_[1:0] / VRL1_[1:0]	29	31	31	22
	VSS1	VSS1	74	57	57	43
	VDDF	VDDF	73	58	58	44
	VDDX1	VDDX1	78	62	62	46
	VSSX1	VSSX1	79	63	63	47
	BCTL	BCTL	9	9	9	5
LIN	LIN0	LIN0[1][2][3]	-	-	1	1
	LGND	LGND[1][2][3]	-	-	64	48
CAN VREG	BCTLC	BTCLC[4][5]	37	1	-	-
	VDDC	VDDC[4][5]	38	64	-	-
CANPHY	CANH0	CANH0	39	-	-	-
	VSSC	VSSC	40	-	-	-
	CANL0	CANL0	41	-	-	-
	SPLIT0	SPLIT0	42	-	-	-
VREG S1	SNPS1	SNPS1	10	-	-	-
	BCTLS1	BCTLS1	11	-	-	-
	VDDS1	VDH0_1 / VRH1_1	12	-	-	-
VREG S2	SNPS2	SNPS2	13	-	-	-
	BCTLS2	BCTLS2	14	-	-	-
	VDDS2	VRH0_2 / VRH1_2	15	-	-	-
E	PE1	PTE[1] / XTAL	66	55	55	41
	PE0	PTE[0] / EXTAL	67	56	56	42

Table continues on the next page...

[1] Available for S12ZVML128/64/32
[2] Available for S12ZVML31
[3] Available for S12ZVM32/16
[4] Available for S12ZVMC256
[5] Available for S12ZVMC128/64

Table 2. Pin differences between packages of S12ZVM devices (continued)

Module	Module pin	Pin function	Pin # in 80 LQFP	Pin # in 64C LQFP	Pin # in 64L LQFP	Pin # in 48 LQFP
AD	PAD15	PTADH[7] / KWADH[7] / AN0_7 / PDOCLK	36	-	-	-
	PAD14	PTADH[6] / KWADH[6] / AN0_6 / PDO	35	-	-	-
	PAD13	PTADH[5] / KWADH[6] / AN0_5 / PTURE	34	-	-	-
	PAD12	PTADH[4] / KWADH[4] / AN1_7	33	-	-	-
	PAD11	PTADH[3] / KWADH[3] / AN1_6	32	-	-	-
	PAD10	PTADH[2] / KWADH[2] / AN1_5	31	-	-	-
	PAD9	PTADH[1] / KWADH[1] / AN1_4	30	-	-	-
	PAD8	PTADH[0] / KWADH[0] / AN1_3 / VRH[1][2][3][5]	27	29	29	20
	PAD7	PTADL[7] / KWADL[7] / AN1_2 / AMPP1	26	28	28	-
	PAD6	PTADL[6] / KWADL[6] / AN1_1 / $\overline{\text{SS0}}$ / AMPM1	25	27	27	-
	PAD5	PTADL[5] / KWADL[5] / AN1_0 / AMP1	24	26	26	-
	PAD4	PTADL[4] / KWADL[4] / AN0_4	23	25	25	-
	PAD3	PTADL[3] / KWADL[3] / AN0_3	22	24	24	-
	PAD2	PTADL[2] / KWADL[2] / AN0_2 / AMPP0	21	23	23	19
	PAD1	PTADL[1] / KWADL[1] / AN0_1 / AMPM0	20	22	22	18
	PAD0	PTADL[0] / KWADL[0] / AN0_0 / AMP0	19	21	21	17
T	PT3	PTT[3] / PWM0_3[4] / IOC0_3 / PWM1_2[4] / $\overline{\text{SS0}}$	64	53	53	-
	PT2	PTT[2] / PWM0_7[4] / IOC0_2 / PWM1_0[4]1 / PWM1_5[1][2][3][5] / SCK0	63	52	52	-
	PT1	PTT[1] / IOC0_1 / PWM1_4 / MOSI0 / LPDC0[1][2][3] / TXD0 / PTURE[1][2][3][5]	62	51	51	-
	PT0	PTT[0] / PWM0_5[4] / IOC0_0 / PWM1_3 / MISO0 / RXD0	61	50	50	39

Table continues on the next page...

Table 2. Pin differences between packages of S12ZVM devices (continued)

Module	Module pin	Pin function	Pin # in 80 LQFP	Pin # in 64C LQFP	Pin # in 64L LQFP	Pin # in 48 LQFP
S	PS5	PTS[5] / KWS[5] / $\overline{\text{SS0}}$ / PDO[1][5]	-	8	8	-
	PS4	PTS[4] / KWS[4] / SCK0 / PDOCLK[1][5]	-	7	7	-
	PS3	PTS[3] / KWS[3] / DBGEEV / TXD1 / CPTXD0[4] / IOC1_1[4] / MOSI0	69	6	6	-
	PS2	PTS[2] / KWS[2] / RXD1 / CPRXD0[4] / IOC1_0[4] / MIS00	70	5	5	-
	PS1	PTS[1] / KWS[1] / TXD1 / TXCAN0[1][4][5] / CPDR1[4] / LPTXD0[1][2][3] / IOC0_2[2][3][4] / PTUT1 / SCK0[4]	71	4	4	4
	PS0	PTS[0] / KWS[0] / RXD1 / RXCAN0[1][4][5] / LPRXD0[1][2][3] / IOC0_1[2][3][4] / PTUT0 / $\overline{\text{SS0}}$[4]	72	3	3	3
P	PP2	PTP[2] / KWP[2] / PWM1_2	-	59	59	-
	PP1	PTP[1] / KWP[1] / PWM0_1[4] / PWM1_1 / $\overline{\text{IRQ}}$	76	60	60	-
	PP0	PTP[0] / KWP[0] / EVDD1 / PWM1_5[4] / PWM1_0[1][2][3][5] / ECLK / FAULT5 / $\overline{\text{XIRQ}}$	77	61	61	45
L	PL0	PTIL[0] / KWL[0]	8	-	-	-

1.3 Mask set differences

The parts S12ZVML128, S12ZVMC128, S12ZVML64, S12ZVMC64, and S12ZVML31 have the following mask set options. The gray shaded columns are the non-production mask sets.

Table 3. N95G mask set option table

	ZVMx128, ZVMx64, ZVML32				ZVML31, ZVM32, ZVM16		ZVMC256	
Feature	**0N95G**	**1N95G**	**2N95G**	**3N95G**	**0N14N**	**1N14N**	**0N00R**	**1N00R**
LINPHY supply pin	HD	HD	VSUP	HD	HD	HD	-	-
BST pin function available	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
GDU low side driver state in HD over-voltage case	On	GOCA1	GOCA1	GOCA1		GOCA1		GOCA1

Table continues on the next page...

Table 3. N95G mask set option table (continued)

	ZVMx128, ZVMx64, ZVML32				ZVML31, ZVM32, ZVM16		ZVMC256	
GDU HD nominal over-voltage time constant	300 ns	300 ns	2.7 µs	2.7 µs	2.7 µs	2.7 µs	2.7 µs	2.7 µs
GDU GSUF bit state one clock cycle after reset	1	1	0	FOPT:NV[6]	1	$\overline{\text{FOPT:NV[7]}}$	$\overline{\text{FOPT:NV[7]}}$	$\overline{\text{FOPT:NV[7]}}$
EPRES (GDUE[5]) Inclusion	Not usable	Not usable	Not usable	Not included	Not included	Not included	Not included	Not included
GDUCTR1 Available bits	None	None	None	GDUCTR1 [0]	None	None	GDUCTR1 [7,6,0]	GDUCTR1 [7,6,0]

CAUTION

The mask set 2N95G uses the VSUP pin as the LINPHY supply. Thus the **BST function must not be used on this mask set** because enabling it could cause a LINPHY supply voltage offset with respect to other devices on the LIN bus.

Table 4. List of special mask set differences

Mask Set	Parameter	Standard	Mask Set Specific
N06E	GDU Module Register Address	0x06A0 – 0x06BF	0x0690 – 0x069F

2 Application overview

The S12ZVM devices are designed for a 3-phase motor control and optimized to cover BLDC, PMSM, and SR motor control applications with a reduced set of external components. A typical application is shown in Figure 1. S12ZVM application overview on page 8.

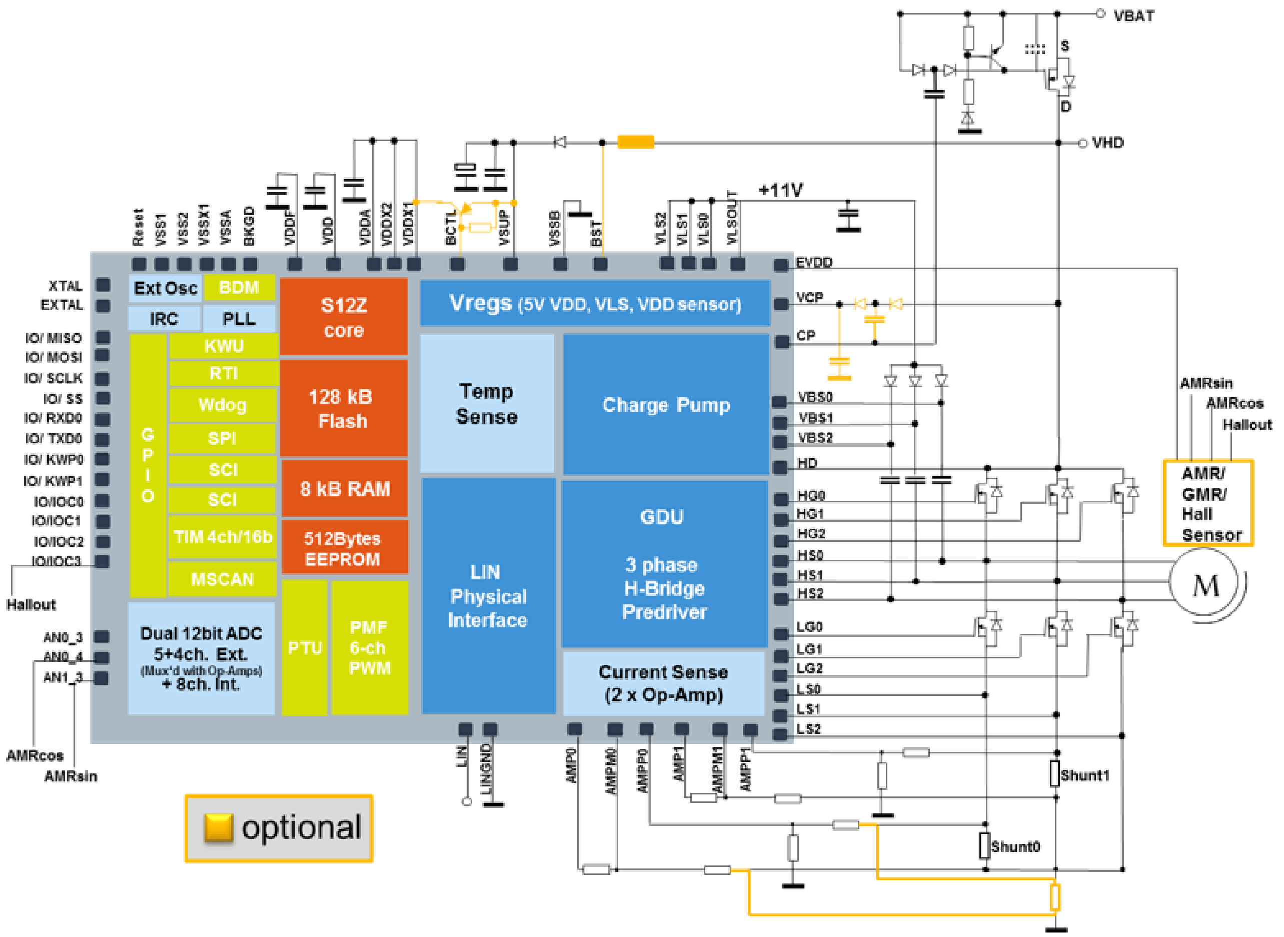


Figure 1. S12ZVM application overview

The main reason for migrating the application is usually to extend the feature set or, more often, to reduce the MCU cost and/ or PCB size. Users can also migrate their development kit based application to the production hardware, where a different package or mask set is used.

It is highly recommended to consider all possible hardware and software changes before the design of the new application starts. Some changes are obvious, such as migrating from 64-pin package to 48-pin package, which enables the single-shunt current sensing only, while the 64-pin package enables both dual-shunt and single-shunt solution. However, several changes are necessary to be considered due to a different mask set or a different versions of the internal peripheral modules. This migration guide lists the differences and suggests the changes needed to keep the same functionality where possible. Some useful examples are introduced in chapters Migration of PMSM single-shunt application (AN5327_SW) from S12ZVML128 to S12ZVML31 on page 17 and Migration of the PMSM single-shunt application (AN5327_SW) code to a custom application on page 29.

3 Module versions and differences

Table 5. S12ZVM module version table on page 9 provides a summary of module version differences within the MC9S12ZVM-Family. The differences between the module versions are summarized in the individual module chapters. Modules which are not listed in this table have identical versions and features across all MC9S12ZVM-Family members.

Table 5. S12ZVM module version table

Feature	ZVMC256	ZVML128	ZVMC128	ZVML64	ZVMC64	ZVML32	ZVML31	ZVM32	ZVM16
PIM	V3	V2	V2	V2	V2	V2	V2	V2	V2
CPMU	V10	V6	V6	V6	V6	V6	V6	V6	V6
PMF	V4	V3	V3	V3	V3	V3	V4	V4	V4
GDU	V6	V4	V4	V4	V4	V4	V5	V5	V5
DBG	V4	V2	V2	V2	V2	V2	V3 (Lite)	V3 (Lite)	V3 (Lite)
ADC	V3	V1	V1	V1	V1	V1	V1	V1	V1

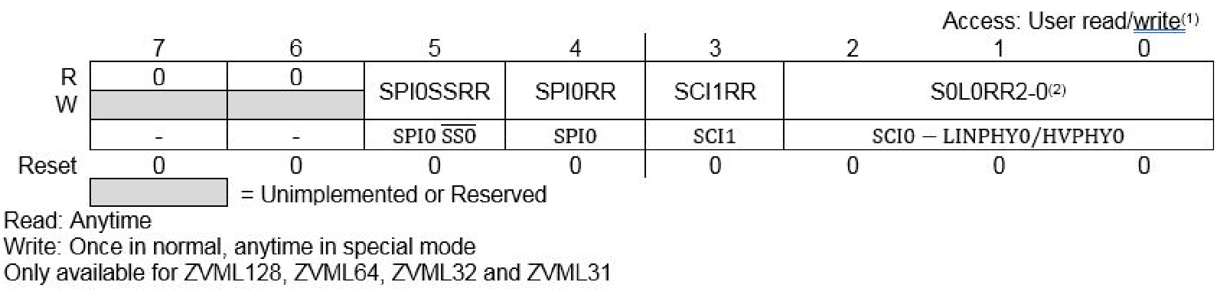
3.1 Port Integration Module (PIM)

The S12ZVM family introduces V2 and V3 versions of the PIM. When migrating the application, users should take into consideration a usage of a different module or ports. These differences are listed in Table 2. Pin differences between packages of S12ZVM devices on page 3. There are several differences in the module routing options which are listed below. Migration between ZVMC256 and other devices requires special attention to the CAN/LIN modules and related peripheral pins. For more details, please see the S12ZVM Reference Manual available at nxp.com.

3.1.1 Module Routing Register 0 (MODRR0)

The MODRR0 introduces S0L0RR[2:0] bits which are available for ZVML devices only. In case of migration from ZVMC devices to ZVML, no action is required since the default settings connect the RXD and TXD to the LINPHY module. When migrating from ZVML devices, reading and writing to these bits need to be removed.

The SPI0RR bit controls the routing of the SPI0 module. The S12ZVMC256 device has the SCK0 routed to PS1 pin instead of PS4, and the $\overline{SS0}$ routed to PS0 pin instead of PS5. Therefore, special attention should be taken to the hardware connection changes when migrating to or from the S12ZVMC256.


Access: User read/write(1)

	7	6	5	4	3	2	1	0
R	0	0	SPI0SSRR	SPI0RR	SCI1RR	S0L0RR2-0(2)		
W	▨	▨						
	-	-	SPI0 $\overline{SS0}$	SPI0	SCI1	SCI0 – LINPHY0/HVPHY0		
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

1. Read: Anytime
Write: Once in normal, anytime in special mode
2. Only available for ZVML128, ZVML64, ZVML32 and ZVML31

Figure 2. Module Routing Register 0 (MODRR0)

3.1.2 Module Routing Register 1 (MODRR1)

The first three bits of the MODRR1 are available only for S12ZVMC256, thus any access to these bits need to be removed when migrating to the other devices. The default setting routes TXCAN and RXCAN to the CANPHY0 module.

Access: User read/write[1]

	7	6	5	4	3	2	1	0
R W	M0C0RR2-0[2]			PWMPRR1-0[3]		PWM54RR	PWM32RR	PWM10RR
	MSCAN0-CANPHY0 interface			PWM probe		PWM1_4 PWM1_5 GDU/pins	PWM1_2 PWM1_3 GDU/pins	PWM1_1 PWM1_0 GDU/pins
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

1. Read: Anytime
   Write: Once in normal, anytime in special mode
2. Only available for ZVMC256
3. PWMPRR[1] only writable for ZVMC256

Figure 3. Module Routing Register 1 (MODRR1)

PWM signals can be probed on related external pins if enabled in PWMPRR1-0. The bit PWMPRR1 is writable only for S12ZVMC256, thus changes to the code should be performed to disable writing to this pin when migrating to the other devices.

Output PWM signals can be routed to the GDU module or to the external pins. For S12ZVMC256, different external pins are used and should be considered at hardware design time (Table 6. PWM external pin routing differences of S12ZVM family on page 10).

Table 6. PWM external pin routing differences of S12ZVM family

PWM output	S12ZVMC256 pin	Other S12ZVM devices pin
PWM1_0	PT2	PP0
PWM1_1	PP1	PP1
PWM1_2	PT3	PP2
PWM1_3	PT0	PT0
PWM1_4	PT1	PT1
PWM1_5	PP0	PT2

3.1.3 Module Routing Register 2 (MODRR2)

MODRR2 sets routing of timer input-output capture signals to the external pins or other modules. The S12ZVMC256 introduces new features and settings in the MODRR2.

TIM0 IC1, IC0_1 and IC0_2 signal routing options are available only for ZVMC256, ZVML31, ZVM32 and ZVM16. The IC1 signal can be used to determine the asynchronous commutation event in BLDC motor applications with Hall sensors. The IC0_2 signal can be used to detect HVI signal on S12ZVMC256 device only. Corresponding settings of T0C2RR1-0, T0C1RR and T0IC1RR0 bits should be used only on ZVMC256, ZVML31, ZVM32 and ZVM16 devices. When migrating from other devices, the default settings are applied if no change to code is made.

The T0IC1RR0 bit overrides the T0IC1RR and T0C1RR settings. When migrating from the devices mentioned above, the bit setting should be removed.

T1IC0RR bit setting enables the GDU delay measurement (t_{delon}) on S12ZVMC256 devices.

Access: User read/write(1)

	7	6	5	4	3	2	1	0
R W	T0C2RR1-0(2)		T0C1RR(2)	T1IC0RR(3)	T0IC3RR1-0		T0IC1RR	T0IC1RR0(2)
	IOC0_2		IOC0_1	IC1_0	TIM0 IC3		TIM0 IC1	TIM0 IC1
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

1. Read: Anytime
   Write: Once in normal, anytime in special mode
2. Only available for ZVMC256, ZVML31, ZVM32 and ZVM16
3. Only available for ZVMC256

Figure 4. Module Routing Register 2 (MODRR2)

3.2 Clock, Reset and Power Management Unit (CPMU)

There are two versions V10 and V6 of the CPMU module in the S12ZVM devices, where the V10 version is linked to the S12ZVMC256 devices. Thus, the migration process should consider changes only if migrating from S12ZVMC256 to other S12ZVM device. For more details, please see the S12ZVM Reference Manual available at nxp.com.

The differences are connected mainly with the external voltage regulators for VDDS1 and VDDS2 domains. When migrating applications from other S12ZVM to S12ZVMC256, no action is required since the after-reset default values of the S12ZVMC256 specific bits are zeros. Users should avoid writing these bits when migrating from S12ZVMC256 to other S12ZVM devices. The below figures shows the registers with extended functionality of the S12ZVMC256 device.

	7	6	5	4	3	2	1	0
R	VRH2EN1	VRH1EN1	EXTS2ON1	EXTS1ON1	0	EXTCON	EXTXON	INTXON
W					▨			
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

1. Only available in V10

Figure 5. Voltage Regulator Control Register (CPMUVREGCTL)

	7	6	5	4	3	2	1	0
R	0	0	0	0	VDDSIE1	LVDS	LVIE	LVIF
W	▨	▨	▨	▨		▨		
Reset	0	0	0	0	0	U	0	U

▨ = Unimplemented or Reserved

1. Only available in V10

Figure 6. Low Voltage Control Register (CPMULVCTL)

	7	6	5	4	3	2	1	0
R	SCS2	SCS1	LVDS2	LVDS1	SCS2IF	SCS1IF	LVS2IF	LVS1IF
W	▨	▨	▨	▨				
Reset	0	0	0	0	0	U	0	U

▨ = Unimplemented or Reserved

Figure 7. VDDS Status Register (CPMUVDDS) – available only in CPMU_V10

3.3 Pulse Width Modulator with fault protection (PMF)

The main differences between V3 and V4 versions of PMF module of the S12ZVM devices are listed in the below table.

Table 7. PMF module versions differences

Feature	V3	V4
Write protection (WP) on REV1-0 bits	Not available	Available
Ability to read the PWM output value through PMFOUTB register	Not available	Available

The REV1-0 bits select if timebase generator A, B or C provides the reload event on output signal pmf_reloada. However, write protection is available in V4 only.

The PMFOUTB register provides access to the software control of the outputs if the corresponding OUTCTLn bit is set. Reading the register in V3 returns the register value. In V4, the current output states are returned.

NOTE

The S12ZVMC256 devices feature additional Pulse Width Modulation Module (PWM) with 8x 8bit channels or 4x 16bit channels. It is not possible to migrate the PWM code from S12ZVMC256 to other devices. For more information, please see the S12ZVM Reference Manual available at nxp.com.

3.4 Timer (TIM1) module

The TIM1 module is introduced on S12ZVMC256 devices only. When migrating an application from S12ZVMC256, all the TIM1 references and calls should be removed and the functionality should be replaced by TIM0 module or by software.

3.5 CAN Physical Layer (CANPHY)

The CAN physical layer is featured on S12ZVMC256 only. When migrating from S12ZVMC256 to other ZVMC devices, the MODRR1 should be changed to connect the RXCAN and TXCAN signals to the output pins. No action is needed when migrating to the S12ZVMC256 device, however the CANPHY module can be used to reduce PCB design requirements.

3.6 Gate Drive Unit (GDU)

The differences between V4, V5 and V6 versions of GDU module are listed in the Table 8. GDU module versions differences on page 13. Description of each row is presented in the next paragraphs.

[6] This migration guide provides an example of the migration from the S12ZVML12, which features the GDUV4 (orange shaded column), and to the S12ZVML31, which features the GDUV5 (green shaded column). For more

Table 8. GDU module versions differences

Feature	V4[6]	V5[6]	V6
TDEL control bit for tdelon/tdeloff on page 13	not available	not available	available
Number of overcurrent threshold bits for overcurrent comparator 0/1 on page 13	GOCT0[3:0] GOCT1[3:0]	GOCT0[4:0] GOCT1[4:0]	GOCT0[4:0] GOCT1[4:0]
VLS level select control bit GVLSLVL on page 14	not available	available	available
Current sense amplifier offset on page 14	adjustable in 5mV steps	adjustable in 3mV steps	adjustable in 3mV steps
On chip bootstrap diode on page 14	not available, off chip bootstrap diode required	available	not available, off chip bootstrap diode required
Desaturation filter bits GDSFLS/GDSFHS on page 14	not available	available	available
Fault[3] output to PMF on page 14	driven by GLVLSIF	driven by GLVLSF	driven by GLVLSF
Fault[4] output to PMF on page 15	driven by GHHDIF	driven by GHHDF	driven by GHHDF
Low-side drivers on or off out of reset dependent on NVM option on page 15	not available	not available[7]	available
Additional drain connections LD[2:0] to external low-side power FETs on page 15	not available	not available	available
Control bits GSRMOD1 and GSRMOD0 for SR motor drive on page 15	not available	not available	available

3.6.1 TDEL control bit for t_{delon}/t_{deloff}

The TDEL bit of GDUCTR1 register controls the parameters t_{delon} and t_{deloff}. The GDUV6 introduces a specific t_{delon} and t_{deloff} propagation times if the TDEL bit is set. Migrating the application from GDUV4 or GDUV5 to GDUV6 has no effect on the performance, while migrating from GDUV6 to other versions of GDU affects the propagation delay, if TDEL was set within the original application. For more information, please see the S12ZVM Reference Manual available at nxp.com.

3.6.2 Number of overcurrent threshold bits for overcurrent comparator 0/1

The overcurrent comparator threshold voltage is the output of a 6-bit digital-to-analog converter. The upper two bits of the digital inputs are tied to one. The other bits of the digital inputs are driven by GOCTX (GOCT0 and GOCT1 respectively). The overcurrent comparator threshold voltage can be calculated from equations below, Equation 1 for GDUV4 and Equation 2 for GDUV5 and V6.

Equation1

$$\mathrm{VoctX} = (48 + \mathrm{GOTX}).\frac{\mathrm{VDDA}}{64}$$

Equation2

details, see the example in Migration of PMSM single-shunt application (AN5327_SW) from S12ZVML128 to S12ZVML31 on page 17.

[7] See device overview for maskset / GDU version information in S12ZVM Reference Manual available at nxp.com.

$$VoctX = (32 + GOTX).\frac{VDDA}{64}$$

For the GDUV4, upper two bits are tied to one and for the GDUV5 and V6 the upper one bit is tied to one. When migrating the application, following equations should be used to keep the threshold on the same level. When migrating from GDUV4 to a higher version, the change is valid for the whole range of original values <0, 15>.

Equation3

$$GOCTX_{GDUV5,GDUV6} = GOCTX_{GDUV4} + 16$$

When migrating from GDUV5 or GDUV6 to the version GDUV4, the GOCTX input range is limited by the valid range of the GDUV4, thus the valid range of $GOCTX_{GDUV5,\ GDUV6}$ is <16, 31>.

Equation4

$$GOCTX_{GDUV4} = GOCTX_{GDUV5,GDUV6} - 16$$

3.6.3 VLS level select control bit GVLSLVL

The GVLSLVL bit of the GDUCTR register selects the voltage threshold of the undervoltage detection on VLS pin. The GDUV4 acts as if the GVLSLVL is set, therefore the low voltage monitor is detecting V_{LVLSHA} threshold value. If the original application uses V_{LVLSHA}, no action is needed. Nevertheless, if the original application uses the VLVLSLA level, the destination application should consider changing the level to the V_{LVLSHA}.

3.6.4 Current sense amplifier offset

The GCSOx[2:0] bits of the GDUCSO register set the offsets of the current sense amplifiers. The offset is adjustable with a step of 5 mV for GDUV4 and a step of 3 mV for GDUV5 and V6. When migrating from V4 to V5 or V6 version of GDU, the GDUCSO register should be adjusted respectively.

3.6.5 On chip bootstrap diode

The GDUV4 and V6 has no bootstrap diode implemented. When migrating from GDUV4 or GDUV6 to GDU V5 the hardware design of the destination application should consider the GDUV5 has the bootstrap diode implemented. Migrating from GDUV5 to the other two versions, the bootstrap diodes should be incorporated.

3.6.6 Desaturation filter bits GDSFLS/GDSFHS

The desaturation bits GDSFLS and GDSFHS of the GDUDSLVL register (GDUV5 and GDUV6) adjust the desaturation filter characteristics of the three low-side and three high-side FET pre-drivers. Migrating from GDUV4 to GDUV5 or GDUV6 does not require any action, however it is possible to adjust the filter according to the S12ZVM Reference Manual available at nxp.com. When migrating the application using GDUV5 or GDUV6 to the application using GDUV4, the desaturation comparator output is not filtered and setting the GDSFLS and GDSFHS bits of the GDUDSLVL register has no effect.

3.6.7 Fault[3] output to PMF

In GDUV4, the Fault[3] signal is driven by the GLVLSIF bit (GDUF register), which is set by hardware if GLVLSF bit is set or GLVLSS (GDUSTAT register) is cleared. These bits are changed simultaneously based on the VLS_OUT Supply voltage status compared to the V_{LVLSHA}, V_{LVLSLA} and V_{LVLSD}. In GDUV5 and V6, the Fault[3] signal is driven by the GLVLSF bit directly, which allows to clear the interrupt flag and the fault flag independently. If the fault protection logic is used in the original application, the clear-fault logic should be changed in the destination application. To clear the fault, GLVLSF bit should cleared by writing 1 instead of GLVLSIF. However, if the application uses low VLS supply interrupt routine, clearing the GLVLSIF should be handled as well.

3.6.8 Fault[4] output to PMF

On GDUV4, the Fault[4] signal is driven by the GHHDIF bit (GDUF register), which is set by hardware if GHHDF bit is set or GHHDS (GDUSTAT register) is cleared. These bits are changed simultaneously based on the HD pin voltage status compared to the V_{HVHDLD}, V_{HVHDHD}, V_{HVHDLA} and V_{HVHDHA}. In GDUV5 and V6, the Fault[4] signal is driven by the GHHDF bit directly, which allows to clear the interrupt flag and the fault flag independently. If the fault protection logic is used in the original application, the clear-fault logic should be changed in the destination application. To clear the fault, GHHDF bit should cleared by writing 1 instead of GHHDIF. However, if the application uses high VHD supply interrupt routine, clearing the GHHDIF should be handled as well.

3.6.9 Low-side drivers on or off out of reset dependent on NVM option

On GDUV4 the startup flag GSUF of the GDUF register is cleared by reset and set by HW after reset deasserts. On GDUV5 and V6 the flag is cleared by reset and loaded from flash option field after reset deasserts. Refer also to S12ZVM Reference Manual – chapter 1.12 Module device level dependencies and chapter 1.2.3 Functional differences between masksets. If the original application doesn't use the flash option field (the NV[7] is cleared by default), the GSUF is set, thus acts the same way as the GDUV4 version (the application usually writes 1 to the GSUF to clear the start-up status). The GSUF behavior can be changed though.

When migrating from the application using the flash option to the application involving GDUV4, the default behavior is expected as if the NV[7] bit is cleared. Thus, the GSUF should be handled by the user application on every reset.

3.6.10 Additional drain connections LD[2:0] to external low-side power FETs

Additional drain connections are available on GDUV6 to enable SR motors to be driven. GSRMOD1 bit of the GDUCTR1 register should be cleared to keep the desaturation comparator connected to the HSx pin for the PMSM/BLDC application. Since other versions of GDU don't feature SR motor enablement, it is assumed these register bits are set to default (zero) or not used when migrating the application. Thus there is no action needed.

3.6.11 Control bits GSRMOD1 and GSRMOD0 for SR motor drive

The GDUCTR1 register of the GDUV6 version enables the SR motor application to be driven by S12ZVM device. The S12ZVMC256 is the only device equipped with GDUV6 version. When migrating the application from GDUV4 to GDUV6, no action is needed since these bits are zero by default. When migrating from GDUV6 to GDUV4, the GDUCTR1 register is not available, thus the setting should be removed from the code.

3.7 Debug Module (DBG)

Table 9. Comparison of S12Z Debug module versions on page 15 shows the comparison of S12Z Debug module versions. When migrating from V2 or V4 DBG to V3 (Lite) version, these debug limitations should be considered when debugging the application.

Table 9. Comparison of S12Z Debug module versions

S12Z Debug V2	S12Z Debug V4	S12Z Debug V3 (Lite)
Tracing included	Tracing included	Tracing not included

Table continues on the next page...

Table 9. Comparison of S12Z Debug module versions (continued)

S12Z Debug V2	S12Z Debug V4	S12Z Debug V3 (Lite)
Profiling included	Profiling included	Profiling not included
Comparator C included	Comparator C included	Comparator C not included
Match 2 trigger included	Match 2 trigger included	Match 2 trigger not included
PREND bit not included	PREND bit included	PREND bit not included

3.8 Analog to digital converter (ADC)

Table 10. Comparison of ADC12B_LBA module versions on page 16 introduces the main differences between the versions of ADC modules. The sections below discuss these differences in detail

Table 10. Comparison of ADC12B_LBA module versions

Feature	V1	V2	V3
ADC Command Register 0 (ADCCMD_0): OPT[1:0] bits ADC Command Register 2 (ADCCMD_2): OPT[3:2] bits	No	Yes	Yes
ADC Command Register 1 (ADCCMD_1): VRH_SEL[1:0]	No	No	Yes
ADC Command Register 1 (ADCCMD_1): VRH_SEL, VRL_SEL	Yes	Yes	No
Internal_5 channel usage	Reserved	Reserved	HVI[0]

3.8.1 Option bits OPT[3:0]

These four option bits can be used to control a SoC level feature/function. These bits are placed in the Option bits OPT[1:0] of the ADCCMD_0 and OPT[3:2] of the ADCCMD_2. Please see the device reference manual for details of the feature/ functionality controlled by these bits. When migrating from ADC V3 or V2 to the ADC V1 version, writing to these bits should be avoided. Migrating to the version ADC V3 or V2 from ADC V1 doesn't require any action.

3.8.2 Reference voltage selection bits

These bits select the high/low voltage reference for current conversion, depending on the version of the ADC module.

	23	22	21	20	19	18	17	16
R W	VRH_SEL[1]	VRL_SEL[1]	CH_SEL[5:0]					
R W	VRH_SEL[1:0][2]		CH_SEL[5:0]					
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

1. Only available on ADC12B_LBA V1 and V2 (see Table 10 for details)
2. Only available on ADC12B_LBA V3 (see Table 10 for details)

Figure 8. ADC command register 1 (ADCCMD_1)

Migrating an application from ADC V1 or ADC V2 to ADC V3 and vice versa would require to change the VRxSEL bits according to the below table. This change is highly recommended since the VRL_SEL bit of the ADC V1 and V2 selects low voltage reference and the VRH_SEL[0] bit of the ADC V3 (at the same position within the register) refers to the high voltage reference selection. The VRL_SEL bit name is also excluded from the mc9s12zvmc256.h file provided by NXP.

Table 11. ADCCMD_1 VRx_SEL related field descriptions

Field	Description
	ADC12B_LBA V1 and V2 (includes VRH_SEL/VRL_SEL)
23 VRH_SEL	**Reference high voltage select bit** — This bit selects the high voltage reference for current conversion. 0 VRH_0 input selected as high voltage reference. 1 VRH_1 input selected as high voltage reference.
22 VRL_SEL	**Reference low voltage select bit** — This bit selects the low voltage reference for current conversion. 0 VRL_0 input selected as low voltage reference. 1 VRL_1 input selected as low voltage reference.
	ADC12B_LBA V3 (includes VRH_SEL[1:0])
23-22 VRH_SEL	**Reference high voltage select bit** — These bits select the high voltage reference for current conversion. 00 VRH_0 input selected as high voltage reference 01 VRH_1 input selected as high voltage reference 10 VRH_2 input selected as high voltage reference 11 Reserved

3.8.3 Internal_5 channel usage

The S12ZVMC256 devices have the Internal_5 channel of the ADC connected to HVI[0] pin (general purpose high-voltage pin), while other S12ZVM devices have the Internal_5 channel reserved. There is no action needed when migrating from the other S12ZVM devices to the S12ZVMC256. However, the Internal_5 channel should be removed from the ADC command lists when migrating from S12ZVMC256 to the other S12ZVM devices.

4 Migration of PMSM single-shunt application (AN5327_SW) from S12ZVML128 to S12ZVML31

This chapter introduces the migration of the software AN5327_SW from S12ZVML128 device to S12ZVML31. The application is the PMSM field oriented control with single-shunt current sensing method featuring Motor Control Application Tuning Tool (MCAT). The software is originally designed for MTRCKTSPNZVM128 development kit available at nxp.com/automcdevkits.

4.1 List of MCU differences

Table 12. Differences between S12ZVM development kits on page 18 shows the main differences between the S12ZVMx12EVB and S12ZVM32EVB MCUs including the impact on the code migration.

Table 12. Differences between S12ZVM development kits

Group	EVB,revision	S12ZVMx12EVB,REV D and higher ORIGIN	S12ZVM32EVB, REV B and higher DESTINATION	Change required
MCU	Part No.	S12ZVML12MKH	**S12ZVML31VKH**	
	Mask set	1N95G	**1N14N**	
	Package	64-pin LQFP	64-pin LQFP	
PIM	Version	V2	V2	
CPMU	Version	V6	V6	
PMF	Version	V3	**V4**	
	Write protection (WP) on REV1-0 bits	Not available	**Available**	**No**
	Ability to read the PWM output value through PMFOUTB register	Not available	**Available**	**No**
GDU	Version	V4	**V5**	
	Number of overcurrent threshold bits for overcurrent comparator 0/1 on page 13	GOCT0[3:0] GOCT1[3:0]	**GOCT0[4:0] GOCT1[4:0]**	**No**
	VLS level select control bit GVLSLVL on page 14	not available	**available**	**No**
	Current sense amplifier offset on page 14	adjustable in 5mV steps	**adjustable in 3mV steps**	**No**
	On chip bootstrap diode on page 14	not available, off chip bootstrap diode required	**available**	**Done**
	Desaturation filter bits GDSFLS/GDSFHS on page 14	not available	**available**	**No**
	Fault[3] output to PMF on page 14	driven by GLVLSIF	**driven by GLVLSF**	**No**
	Fault[4] output to PMF on page 15	driven by GHHDIF	**driven by GHHDF**	**No**
DBG	Version	V2	**V3 (Lite)**	**No**
	Tracing	included	**not included**	**No**
	Profiling	included	**not included**	**No**
	Comparator C	included	**not included**	**No**
	Match 2 trigger	included	**not included**	**No**
ADC	Version	V1	V1	
Mask set diff.	GDU HD nominal over-voltage time constant	300ns	**2.7µs**	**No**
	GDU GSUF bit state one clock cycle after reset	1	$\overline{\text{FOPT:NV[7]}}$	**No**
	EPRES (GDUE[5]) Inclusion	Not usable	**Not included**	**No**

4.2 System overview

The original application is based on the S12ZVMLEVBLIN development kit and the AN5327 software. The development kit provides many features, that are not used by the AN5327 software. Thus, the application requires the jumper settings to be set according to the Table 14. Jumper options of the S12ZVMLEVBLIN on page 20.

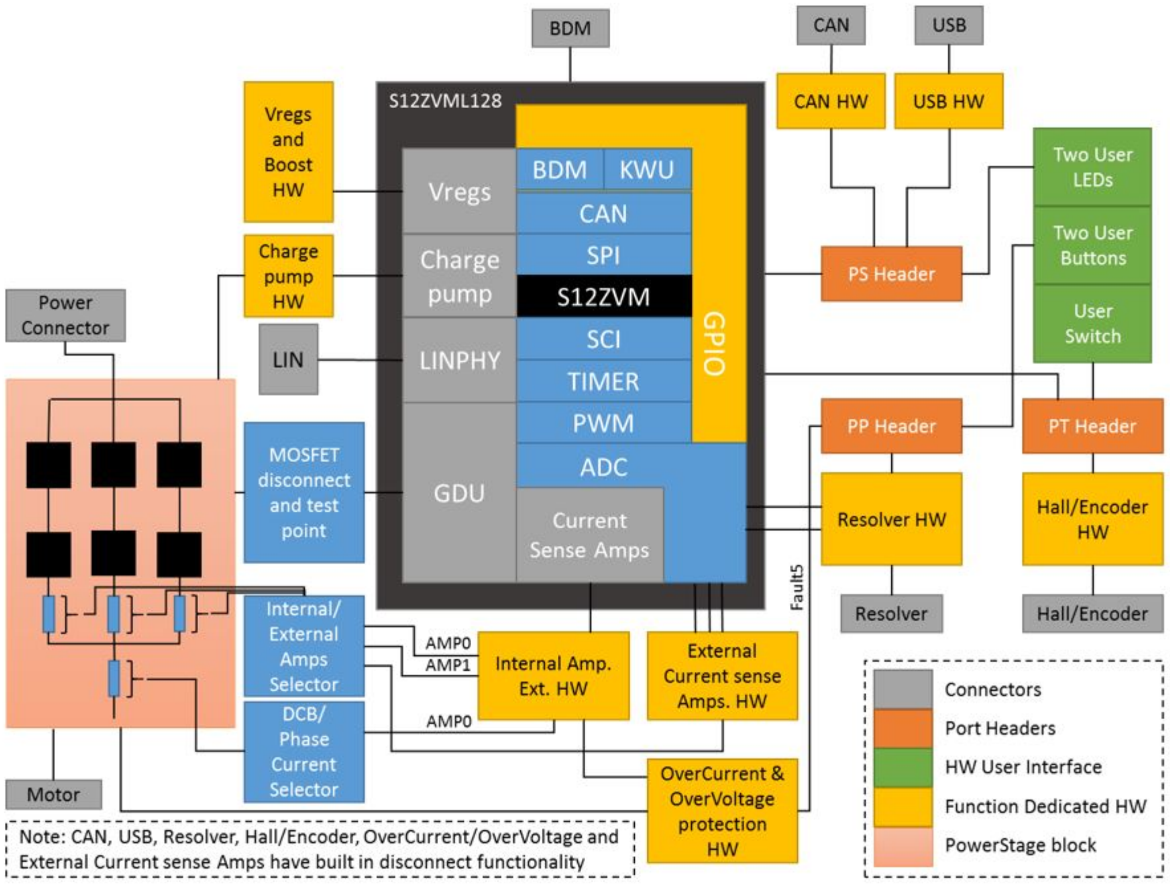


Figure 9. S12ZVMLEVBLIN system overview

The evaluation board can be optionally ordered as S12ZVMx12EVB or S12ZVM32EVB. The PCB is the same, however, some of the components are not used. Table 13. S12ZVMx12EVB vs. S12ZVM32EVB components overview on page 19 provides a list of the component differences.

Table 13. S12ZVMx12EVB vs. S12ZVM32EVB components overview

Item - type	S12ZVMLEVBLIN	S12ZVM32EVB
D9 - BAS16H	Used	Not used (DNP)
D10 - BAS16H	Used	Not used (DNP)
D13 - BAS16H	Used	Not used (DNP)

Table 14. Jumper options of the S12ZVMLEVBLIN

Jumper	Option	Setting	Description
J2	CAN Transceiver 5 V supply option	Open	VDDC ballast transistor is not supplied from VSUP
		Close	VDDC ballast transistor is supplied from VSUP
J3	VDDC supplied from USB option	**Open**	VDDC node is not supplied from the USB-to-SCI interface
		Close	VDDC node is supplied from the USB-to-SCI interface
J4	RESET LED indicator option	Open	RESET LED indicator disabled
		Close	RESET LED indicator enabled
J5	VDDX LED indicator option	Open	VDD LED indicator disabled
		Close	VDD LED indicator enabled
J6	VSUP LED indicator option	Open	VSUP LED indicator disabled
		Close	VSUP LED indicator enabled
J9	ON/OFF switch option	Open	ON/OFF switch disabled
		Close	ON/OFF switch enabled
J10	OSBDM bootloader option	**Open**	OSBDM boot loader disabled
		Close	OSBDM boot loader enabled
J12	Resolver circuit 5 V supply option	Open	Resolver input circuitry (+5VDC) supply disabled
		Close	Resolver input circuitry (+5VDC) supply enabled
J14	EVDD or FAULT selector	1-2	FAULT input is connected to port PP0
		2-3	Port PP0 is connected to EVDD at Hall sensor interface
J15	Resolver or Hall/encoder phase A selector	**1-2**	Phase A from resolver is connected to port PT1
		2-3	Phase A from Hall/encoder interface is connected to port PT1
J16	Resolver or Hall/encoder phase B selector	**1-2**	Phase B from Hall/encoder interface is connected to port PT2
		2-3	Phase B from resolver is connected to port PT2
J18	USB-to-SCI interface supply option	Open	USB-to-SCI isolator supply disable
		Close	USB-to-SCI isolator supply enable
J19	“UP” push button option	Open	“UP” push button disabled
		Close	“UP” push button enabled
J20	“DOWN” push button option	Open	“DOWN” push button disabled
		Close	“DOWN” push button enabled
J27	SCI RXD selector	1-2	RXD from OSBDM is connected to port PS2
		2-3	RXD from USB-to-SCI is connected to port PS2

Table continues on the next page...

Table 14. Jumper options of the S12ZVMLEVBLIN (continued)

Jumper	Option	Setting	Description
J28	SCI TXD selector	1-2	TXD from OSBDM is connected to port PS3
		2-3	TXD from USB-to-SCI is connected to port PS3
J29	BDM interface supply option	Open	BDM supply disabled
		Close	BDM supply enabled
J30	ADC potentiometer pull-up option	Open	POT 1 (ADC potentiometer) supply disabled
		Close	POT 1 (ADC potentiometer) supply enabled
J33	MCU supply option	Open	MCU supply disabled
		Close	MCU supply enabled
J35	ADC mapping PAD0	**1-2**	Connects PAD0 to AMP0 external gain-setting resistors
		2-3	Connects PAD0 to POS_SIN resolver output
J36	VREF generation supply option	**Open**	Disconnects VSUP to supply a regulated voltage at VREF2
		Close	Connects VSUP to supply a regulated voltage at VREF2
J37	USER LED1 option	Open	“User LED1” disabled
		Close	“User LED1” enabled
J38	PDO-PDOCLK	**Open**	PDO-PDOCLK not shorted
		Close	PDO-PDOCLK shorted
J39	Resolver phase B selector	**1-2**	SINCOS I/O connector phase B connected to resolver phase B input
		2-3	POS_COS connected to resolver phase B input
J40	VDDX ballast supply option	Open	VDDX ballast is disconnected
		Close	VDDX ballast is connected
J43	Resolver circuit 12 V supply option	**Open**	Resolver generator circuit supply disconnected
		Close	Resolver generator circuit supply connected
J44	ADC mapping PAD1	Open	PAD1 is disconnected from the AMPM0 external gain-setting resistors
		Close	PAD1 is connected to the AMPM0 external gain-setting resistors
J45	ADC mapping PAD2	Open	PAD2 is disconnected from the AMPP0 external gain-setting resistors
		Close	PAD2 is connected to the AMPP0 external gain-setting resistors
J46	ADC mapping PAD3	**1-2**	Connects PAD3 to ADC_IA (phase A current sense from external opamp)

Table continues on the next page...

Table 14. Jumper options of the S12ZVMLEVBLIN (continued)

Jumper	Option	Setting	Description
		2-3	Connects PAD3 to POS_SIN resolver output
J47	ADC mapping PAD4	**1-2**	Connects PAD4 to ADC_IB (phase B current sense from external opamp)
		2-3	Connects PAD4 to ADC potentiometer POT1
J48	ADC mapping PAD5	**1-2**	Connects PAD5 to AMP1 external gain-setting resistors
		2-3	Connects PAD5 to ADC_IB (phase B current sense from external opamp)
J49	Resolver phase A selector	**1-2**	Resolver phase A connected to SINCOS I/O connector phase A input
		2-3	Resolver phase A connected to resolver POS_SIN Schmitt-Trigger
J50	ADC mapping PAD6	**1-2**	Connects PAD6 to AMPM1 external gain-setting resistors
		2-3	Connects PAD6 to POS_COS resolver output
J51/J42	ADC mapping PAD7	**J51(1-2)**	Connects PAD7 to AMPP1 external gain-setting resistors
		J51(2-3)	Connects PAD7 to POS_SIN resolver output
		J42(1)-J52(2)	Connects PAD7 to ADC potentiometer POT1
J52	ADC mapping PAD8	**1-2**	Connects PAD8 to ADC_IC (phase C current sense from external opamp)
		2-3	
J53	USER LED2 option	Open	“User LED 2” disabled
		Close	“User LED 2” enabled
J55	VREF selector	**1-2**	VREF supplied from VDDX
		2-3	VREF supplied from the VREF2 regulator
J56	Resolver COS reference	**1-2**	Input to POS_COS circuit is from OFFSET1
		2-3	Input to POS_COS circuit is from RES_COS_REF
J57	Internal AMP0 input selector (inverting)	1-2	Connects DC bus to the internal AMP0 inverting input (phase A current sense)
		2-3	Connects ground to the internal AMP0 inverting input (DC bus current sense)
J59	Resolver SIN reference	**1-2**	Input to POS_SIN circuit is from RES_SIN_REF
		2-3	Input to POS_SIN circuit is from OFFSET1
J60	Internal AMP0 input selector (non-inverting)	1-2	Connects phase A to the internal AMP0 noninverting input (phase A current sense)

Table continues on the next page...

Table 14. Jumper options of the S12ZVMLEVBLIN (continued)

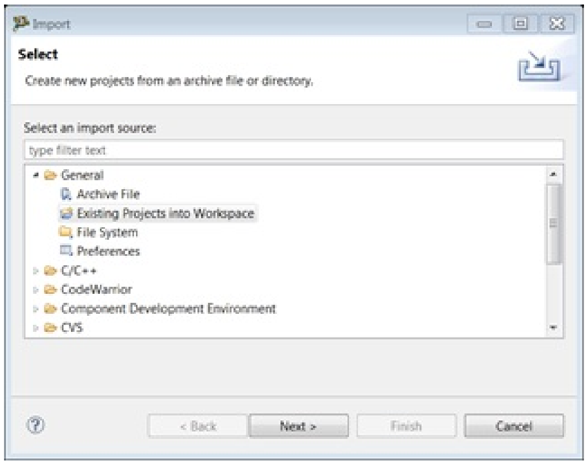
Jumper	Option	Setting	Description
		2-3	Connects DC bus as non-inverting input for internal AMP0 (DC bus current sense)
J63	FAULT comparators 5 V supply option	Open	FAULT circuit supply disconnected
		Close	FAULT circuit supply connected

4.3 Step-by-step software migration

This step-by-step process refers to the migration of AN5327 code, but it can be used respectively to any S12ZVM device code migration. The AN5327software is designed for CodeWarrior 10.6 or higher.

4.3.1 Importing the project

Following steps are mandatory with no specific marker and optional marked as *[Optional:]*. Mandatory steps are required to make the application working on the destination MCU. Nevertheless, the optional steps are recommended as far as they will keep the application settings and definitions consistent with the target MCU. Any of the optional steps requires the rest of the optional steps to be completed.

Action	Details	Screenshot
1) Import the project	Install the AN5327_SW and locate the folder. Then import the project to your workspace. Use the Import wizard, select “General / Existing Projects into Workspace”. In the next step, check the “Copy projects into workspace”.	

Create new projects from an archive file or directory.

Table continues on the next page...

Table continued from the previous page...

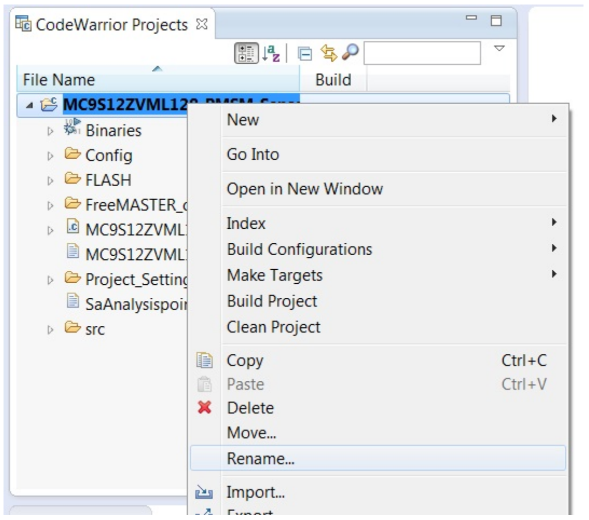
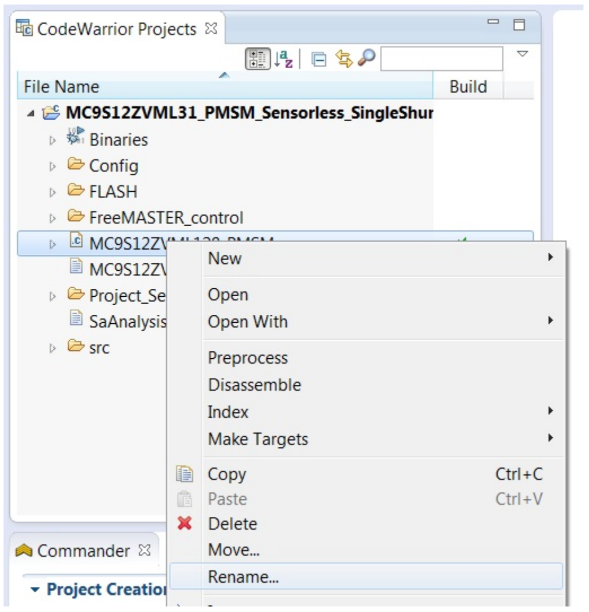
Action	Details	Screenshot
[Optional:] 2) Rename the project	Right-click the project and rename it to S12ZVML31_PMSM_Sensorless_SingleShunt or any other name.	 CodeWarrior Projects File Name Build MC9S12ZVML128_PMSM_Sens Binaries Config FLASH FreeMASTER_c MC9S12ZVML1 MC9S12ZVML1 Project_Setting SaAnalysispoi src New Go Into Open in New Window Index Build Configurations Make Targets Build Project Clean Project Copy Ctrl+C Paste Ctrl+V Delete Move... Rename... Import...
[Optional:] 3) Rename the main file	Right-click the main file “MC9S12ZVML128_PMSM.c” and rename it to “MC9S12ZVML31_PMSM.c”.	 CodeWarrior Projects File Name Build MC9S12ZVML31_PMSM_Sensorless_SingleShur Binaries Config FLASH FreeMASTER_control MC9S12ZVML128_PMSM MC9S12ZV Project_Se SaAnalysis src Commander Project Creatio New Open Open With Preprocess Disassemble Index Make Targets Copy Ctrl+C Paste Ctrl+V Delete Move... Rename...

Table continues on the next page...

Table continued from the previous page...

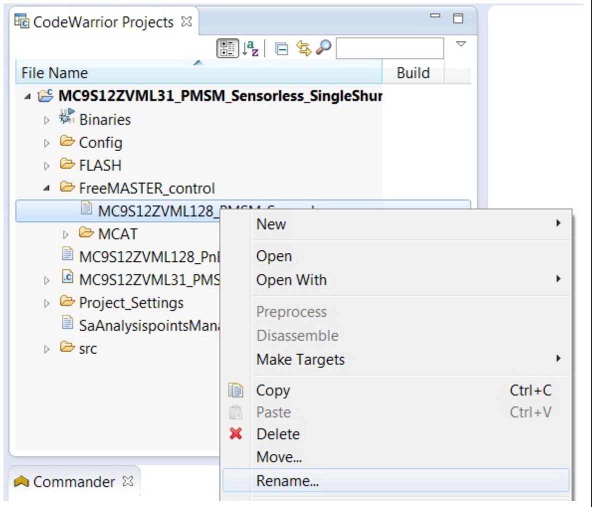
Action	Details	Screenshot
[Optional:] 4) Rename the FreeMASTER project file	Locate the "FreeMASTER_control\ \MC9S12ZVML128_PMSM_Sensorless.pmp and change the name to match the MCU "*ZVML31*".	 CodeWarrior Projects File Name Build MC9S12ZVML31_PMSM_Sensorless_SingleShur Binaries Config FLASH FreeMASTER_control MC9S12ZVML128_ MCAT MC9S12ZVML128_Pn MC9S12ZVML31_PMS Project_Settings SaAnalysispointsMan src Commander New Open Open With Preprocess Disassemble Make Targets Copy Ctrl+C Paste Ctrl+V Delete Move... Rename...
[Optional:] 5) Edit the Analysis Points Manager Path	In CodeWarrior editor or any text editor, open the SaAnalysispointsManager.apconfig and edit the path to the project.	<ProjectPath>*[Path to your workspace]* / MC9S12ZVML128_PMSM_Sensorless_SingleShunt/ SaAnalysispointsManager.apconfig</ProjectPath> to <ProjectPath>*[Path to your workspace]* / MC9S12ZVML31_PMSM_Sensorless_SingleShunt/ SaAnalysispointsManager.apconfig</ProjectPath>
[Optional:] 6) Replace the MCU definition code	Remove (rename to *.old) the files and replace it with the new ones Or Edit and rename the files to *ZVML31*.	mc9s12zvml128.h -> mc9s12zvml31.h in [Project dir]\src\S12ZVM_system\peripherals mc9s12zvml128.c -> mc9s12zvml31.c in [Project dir]\src\S12ZVM_system\startup_CW NOTE: the pinout and HW peripherals are the same, thus the files are identical, except of referencing the ZVML31 instead of ZVML128 device.

Table continues on the next page...

Table continued from the previous page...

Action	Details	Screenshot
[Optional:] 7) Replace the MCU header file references	Check all the header files and C-files affected by the step No.6.	in [Project dir] • MC9S12ZVML31_PMSM.c in [Project dir]\src • actuate_s12zvm.c • actuate_s12zvm.h • bemf_Observer.h • meas_s12zvm.h in [Project dir]\src\S12ZVM_system\peripherals • S12ZVM_system\adc.h • S12ZVM_system\cpmu.h • S12ZVM_system\gdu.h • S12ZVM_system\pim.h • S12ZVM_system\pmf.h • S12ZVM_system\ptu.h • S12ZVM_system\sci.h • mc9s12zvml31.h
8) Edit the command linker file	Edit the command linker file located in [Project dir]\ S12ZVM_system\startup_CW \S12zvm.prm.	Change the memory space setting Change S12ZVML128: RAM = READ_WRITE 0x001000 TO 0x002FFF; To S12ZVML31: RAM = READ_WRITE 0x001000 TO 0x001FFE; Change S12ZVML128: EEPROM = READ_ONLY 0x100000 TO 0x1001FF; To S12ZVML31: EEPROM = READ_ONLY 0x100000 TO 0x10007F; Change S12ZVML128: ROM = READ_ONLY 0xFE0000 TO 0xFFFDFF; To S12ZVML31: ROM = READ_ONLY 0xFF8000 TO 0xFFFDFF;

Table continues on the next page...

Table continued from the previous page...

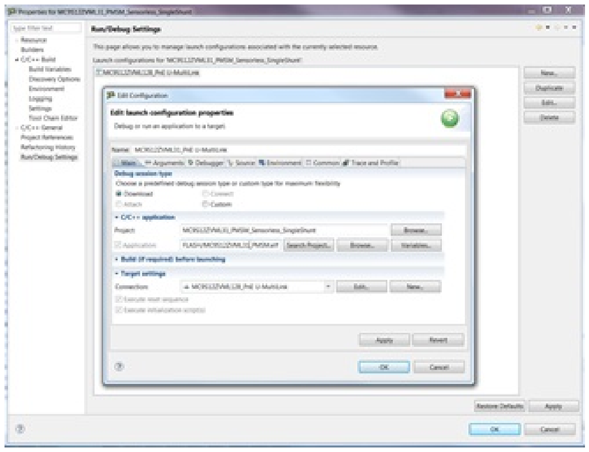
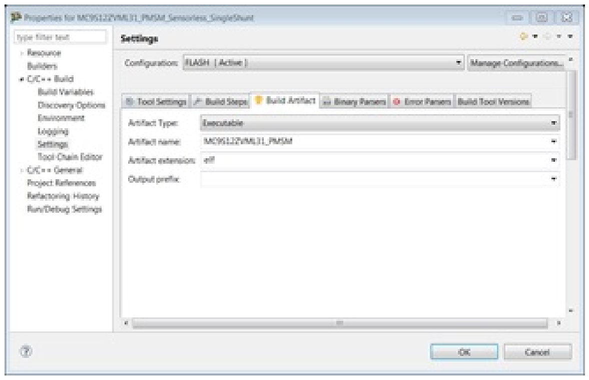
Action	Details	Screenshot
[Optional:] 9) Edit launch configuration	Go to project Properties / “Run/Debug Settings”. Change the name and the Application ELF file name of the C/C++ application.	 Name: MC9S12ZVML31_PnE U-MultiLink Project: MC9S12ZVML31_PMSM_Sensorless_SingleShunt Application: FLASH/MC9S12ZVML31_PMSM.elf Connection: MC9S12ZVML128_PnE U-MultiLink
10) Edit target settings	Go to project Properties / “Run/Debug Settings”. Click “Edit” button next to the Target settings connection list box.	Change the Name of the connection to: MC9S12ZVM32_PnE U-MultiLink Create new target by clicking on “New…” button: Set name to MC9S12ZVM32_PnE U-MultiLink Target Add optional description: Select the Target type to MC9S12ZVM32 Edit Connection type if necessary
[Optional:] 11) Change the Artifact name	Go to project Properties / “Settings”, tab panel “Build Artifact” and change the name to the one specified in the step No. 9.	 Configuration: FLASH [Active] Artifact Type: Executable Artifact name: MC9S12ZVML31_PMSM Artifact extension: elf Output prefix:

Table continues on the next page...

Table continued from the previous page...

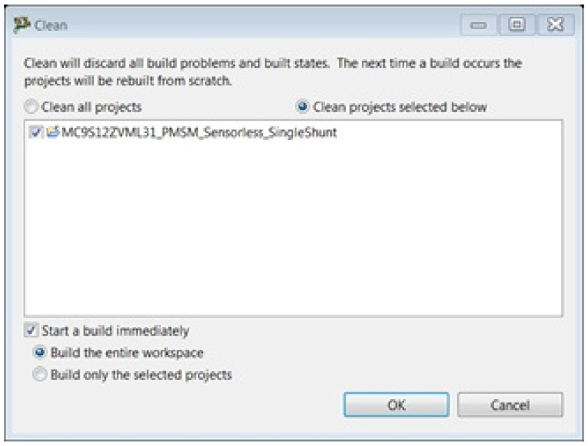
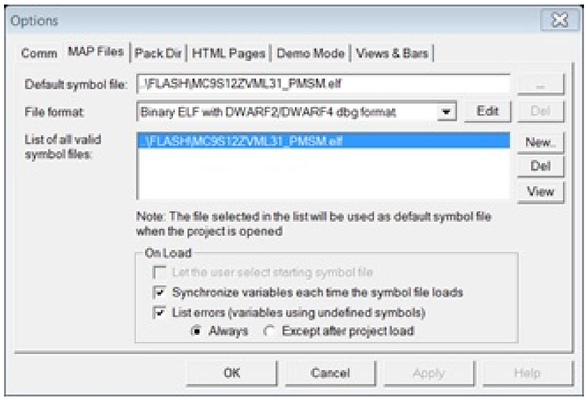
Action	Details	Screenshot
12) Build the project	Go to menu Project / Clean… and check “Start a build immediately” or if not checked, after the project is cleaned, go to menu Project / Build project.	Clean Clean will discard all build problems and built states. The next time a build occurs the projects will be rebuilt from scratch. ○ Clean all projects ● Clean projects selected below ☑ MC9S12ZVML31_PMSM_Sensorless_SingleShunt ☑ Start a build immediately ● Build the entire workspace ○ Build only the selected projects OK Cancel
[Optional:] 12) Change the FreeMASTER map file	Open FreeMASTER project located in [Project dir]\FreeMASTER_control. Open “Project\Options” and edit the “Default symbol file” name located on the MAP Files tab according to previous setting.	Options Comm | MAP Files | Pack Dir | HTML Pages | Demo Mode | Views & Bars Default symbol file: ..\FLASH\MC9S12ZVML31_PMSM.elf File format: Binary ELF with DWARF2/DWARF4 dbg format List of all valid symbol files: ..\FLASH\MC9S12ZVML31_PMSM.elf Note: The file selected in the list will be used as default symbol file when the project is opened On Load ☐ Let the user select starting symbol file ☑ Synchronize variables each time the symbol file loads ☑ List errors (variables using undefined symbols) ● Always ○ Except after project load OK Cancel Apply Help
[Optional:] 13) Update the AMMCLIB	Install the newest version of the AMMCLIB. In project Properties / “Settings”, change the paths to the newest AMMCLIB files.	In S12Z Linker\Input update AMMCLIB version in Libraries list In S12Z Compiler\Access Paths update AMMCLIB version in Search User Paths list In S12Z Assembler\Input update AMMCLIB version in Include File Search Path list

Table continues on the next page...

Table continued from the previous page...

Action	Details	Screenshot
14) Clean and build/run the project	Right click on the project name in the Workspace panel and select "Clear project". Then build and run. In case of any error, please review the steps above and follow the error messages.	

4.3.2 Migrating the software

Table 12. Differences between S12ZVM development kits on page 18 shows that there is no need to change the application software. The GDU low voltage faults and overcurrent faults features can be disabled or adjusted according to the Gate Drive Unit (GDU) on page 12. However, the changes in the configuration files are required, as described in the previous section.

The hardware change needed is to remove the bootstrap diodes, since they are incorporated in the S12ZVML31 chip.

5 Migration of the PMSM single-shunt application (AN5327_SW) code to a custom application

This section describes the most common software migration tasks within the S12ZVM family. All the examples are based on the AN5327_SW software package (PMSM single shunt sensorless field oriented control application).

5.1 Migrating from ADC0 to ADC1

Complexity of the migration process from one ADC module to another, depends on the target application and the way in which the ADC is triggered. If the ADC is triggered by software, the migration is very simple. However, migrating the ADC being triggered by the PTU module needs another PTU to be used due to the hardware connection of ADC0 to PTU0 and ADC1 to PTU1. Therefore, all the settings and registers should be migrated correctly.

The AN5327 software uses both ADC modules. The ADC0 is used to capture the DC link current at four time instants of the PWM period to enable 3-phase current reconstruction. The timing is given by PTU0 triggers, which are calculated based on the double-switching algorithm. The ADC1 measures the internal signal of the DC bus voltage and the internal signal of the MCU's junction temperature.

The S12ZVML128 device features two identical sets of ADC modules and internal operational amplifiers. Since the DC bus voltage and the junction temperature signals are internally connected to both ADC modules, it is possible to swap the functionality of ADC0 and ADC1. Following table shows the steps needed to migrate the signal measurement between ADC0 and ADC1.

Index of changes:

1. Change the PTU trigger list used for 3 phase current reconstruction in actuate_s12zvm.c: SetPtuTriggers()
2. Change the base address of the ADC module from ADC0 to ADC1 in meas_s12zvm.c: GetAdcRawValues(), Change the name of the ADC result list array in meas_s12zvm.c: GetAdcRawValues()

3. Change the name of the ADC result list array in meas_s12zvm.c: Meas_GetUdcVoltage()
4. Change the name of the ADC result list array in meas_s12zvm.c: Meas_GetTemperature()
5. Swap the commands between the ADC0 and ADC1 command lists in adc.c: Definitions, Swap the defaults between the ADC0 and ADC1 results lists in adc.c: Definitions
6. Swap the initial settings between the ADC0 and ADC1 in adc.c: adc0_init()
7. Swap the initial settings between the ADC0 and ADC1 in adc.c: adc1_init()
8. Change the default triggers in ptu.c: Definitions

NOTE

In the following paragraphs, the code layout and comments may be different from the code in the AN5327_SW package in terms of readability and comments. However, the function remains the same.

5.1.1 actuate_s12zvm.c: SetPtuTriggers()

Action

Before

```
void SetPtuTriggers( PTU_TRIGGERS_T       *pTrg)
{
     writeToList = (*(volatile tU8  *)(0x0580 + 0x0006)) & 0x01;
     writeToList  ^= (1 << 0);
     ptuTriggerList0[writeToList][0] = pTrg-> ph1Trg1  +    triggerOffset1;
     ptuTriggerList0[writeToList][1] = pTrg-> ph2Trg1  +    triggerOffset2;

     //ptuTriggerList0[writeToList][2] = pTrg->dcOffsetTrg + triggerOffsetR;

     ptuTriggerList0[writeToList][2] = pTrg-> ph2Trg2  +    triggerOffset3;
     ptuTriggerList0[writeToList][3] = pTrg-> ph1Trg2  +    triggerOffset4;
     ptuTriggerList0[writeToList][4] = 0x00; // End Of List
}
```

After

```
void SetPtuTriggers( PTU_TRIGGERS_T       *pTrg)
{
     writeToList = (*(volatile tU8  *)(0x0580 + 0x0006)) & 0x01;
     writeToList  ^= (1 << 0);
     ptuTriggerList1[writeToList][0] = pTrg-> ph1Trg1  +    triggerOffset1;
     ptuTriggerList1[writeToList][1] = pTrg-> ph2Trg1  +    triggerOffset2;

     //ptuTriggerList1[writeToList][2] = pTrg->dcOffsetTrg + triggerOffsetR;

     ptuTriggerList1[writeToList][2] = pTrg-> ph2Trg2  +    triggerOffset3;
     ptuTriggerList1[writeToList][3] = pTrg-> ph1Trg2  +    triggerOffset4;
     ptuTriggerList1[writeToList][4] = 0x00; // End Of List
}
```

5.1.2 meas_s12zvm.c: GetAdcRawValues()

	Action

Before

```
void GetAdcRawValues(measModule_t *ptr, ADC_RAW_DATA_T *rawData)
{
    volatile tU8 readFromList = 0;

    readFromList = (((*(volatile tU8 *)(0x0600 + 0x0010)))>>6) & 0x01;
    //readFromList  ^= (1 << 0);

    /* removing DC shift of 2.5V ~ OX7FFF  */
    rawData->ph1.f16Arg1    =     (tFrac16)(ADC0ResultList[readFromList][0])
    - (tFrac16)0x7FFF - (tFrac16)ptr->offset.f16Idcb.f16Offset;
    rawData->ph2.f16Arg1    =     (tFrac16)(ADC0ResultList[readFromList][1])
     - (tFrac16)0x7FFF - (tFrac16)ptr->offset.f16Idcb.f16Offset;

    rawData->dcOffset       = 0; //((tFrac32)(ADC0ResultList[readFromList][2])
-

    // (tFrac32)0x7FFF)<<16;

    rawData->ph2.f16Arg2    =     (tFrac16)(ADC0ResultList[readFromList][2])
    - (tFrac16)0x7FFF - (tFrac16)ptr->offset.f16Idcb.f16Offset;
    rawData->ph1.f16Arg2    =     (tFrac16)(ADC0ResultList[readFromList][3])
    - (tFrac16)0x7FFF - (tFrac16)ptr->offset.f16Idcb.f16Offset;
}
```

After

```
void GetAdcRawValues(measModule_t *ptr, ADC_RAW_DATA_T *rawData)
{
    volatile tU8 readFromList = 0;

    readFromList = (((*(volatile tU8 *)(0x0640 + 0x0010)))>>6) & 0x01;
    //readFromList  ^= (1 << 0);

    /* removing DC shift of 2.5V ~ OX7FFF  */
    rawData->ph1.f16Arg1    =     (tFrac16)(ADC1ResultList[readFromList][0])
    - (tFrac16)0x7FFF - (tFrac16)ptr->offset.f16Idcb.f16Offset;
    rawData->ph2.f16Arg1    =
    (tFrac16)( ADC1ResultList[readFromList][1])
     - (tFrac16)0x7FFF - (tFrac16)ptr->offset.f16Idcb.f16Offset;

    rawData->dcOffset       = 0;
    //((tFrac32)(ADC1ResultList[readFromList][2]) - (tFrac32)0x7FFF)<<16;

    rawData->ph2.f16Arg2    =     (tFrac16)(ADC1ResultList[readFromList][2])
    - (tFrac16)0x7FFF - (tFrac16)ptr->offset.f16Idcb.f16Offset;
    rawData->ph1.f16Arg2    =
    (tFrac16)(ADC1ResultList[readFromList][3])
    - (tFrac16)0x7FFF - (tFrac16)ptr->offset.f16Idcb.f16Offset;
}
```

5.1.3 meas_s12zvm.c: Meas_GetUdcVoltage()

Action

Before

```
tBool Meas_GetUdcVoltage(measModule_t *ptr, GDFLIB_FILTER_MA_T *uDcbFilter)
{
    ptr->measured.f16Udcb.raw   = ADC1ResultList[0][0]>>1;
    ptr->measured.f16Udcb.filt  =
        GDFLIB_FilterMA(ptr->measured.f16Udcb.raw, uDcbFilter);

    return(1);
}
```

After

```
tBool Meas_GetUdcVoltage(measModule_t *ptr, GDFLIB_FILTER_MA_T *uDcbFilter)
{
    ptr->measured.f16Udcb.raw   = ADC0ResultList[0][0]>>1;
    ptr->measured.f16Udcb.filt  =
        GDFLIB_FilterMA(ptr->measured.f16Udcb.raw, uDcbFilter);

    return(1);
}
```

5.1.4 meas_s12zvm.c: Meas_GetTemperature()

Action

Before

```
tBool Meas_GetTemperature(measModule_t *ptr)
{
    ptr->measured.f16Temp.raw = (tFrac16)(ADC1ResultList[0][1]);
    ptr->measured.f16Temp.filt = MLIB_Mul_F16(
        ptr->measured.f16Temp.raw,FRAC16(0.73801));
    ptr->measured.f16Temp.filt = MLIB_Sub_F16(
        ptr->measured.f16Temp.filt, FRAC16(0.23801));
    ptr->measured.f16Temp.filt = ptr->measured.f16Temp.filt>>2;
    return(1);
}
```

After

```
tBool Meas_GetTemperature(measModule_t *ptr)
{
    ptr->measured.f16Temp.raw = (tFrac16)(ADC0ResultList[0][1]);
    ptr->measured.f16Temp.filt = MLIB_Mul_F16(
        ptr->measured.f16Temp.raw,FRAC16(0.73801));
    ptr->measured.f16Temp.filt = MLIB_Sub_F16(
        ptr->measured.f16Temp.filt, FRAC16(0.23801));
    ptr->measured.f16Temp.filt = ptr->measured.f16Temp.filt>>2;
    return(1);
}
```

5.1.5 adc.c: Definitions

Action

Before

```
PR_SECTION(adcLists)
volatile char ADC0CommandList[COMMAND_NO][COMMAND_LENGTH] = {
      {0x40,0xD0,0x00,0x00},
// end of sequence [40], current sense channel [D0] - dc bus current on op-amp0
      {0x40,0xD0,0x00,0x00},
// end of sequence [40], current sense channel [D0] - dc bus current on op-amp0
      {0x40,0xD0,0x00,0x00},
// end of sequence [40], current sense channel [D0] - dc bus current on op-amp0
      {0xC0,0xD0,0x00,0x00},
// end of list + no int [C0], current sense channel [D0] - dc bus current on op-amp0
      {0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00}
  };

volatile char ADC1CommandList[COMMAND_NO][COMMAND_LENGTH] = {
      {0x40,0xCB,0x00,0x00},
// end of sequence + no int [40], DC-Link Voltage
      {0xC0,0xC9,0x00,0x00},
// end of List + no int[C0], TEMP [C9], 4clock cycles sample time [00], reserved [00]
      {0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00}
  };

volatile unsigned short ADC0ResultList[2][RESULT_NO] =
{
    {32758, 32758, 32758, 32758, 0, 0, 0, 0},
    {32758, 32758, 32758, 32758, 0, 0, 0, 0}
};
volatile unsigned short ADC1ResultList[2][RESULT_NO] =
{
    {32758, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0}
};
PR_SECTION(DEFAULT_SEC)
```

Table continues on the next page...

Table continued from the previous page...

	Action
After	see code below

```
PR_SECTION(adcLists)
volatile char ADC0CommandList[COMMAND_NO][COMMAND_LENGTH] = {
      {0x40,0xCB,0x00,0x00},
// end of sequence + no int [40], DC-Link VOltage
      {0xC0,0xC9,0x00,0x00},
// end of List + no int[C0], TEMP [C9], 4clock cycles sample time [00], reserved [00]
      {0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00}
  };

volatile char ADC1CommandList[COMMAND_NO][COMMAND_LENGTH] = {
      {0x40,0xD0,0x00,0x00},
// end of sequence [40], current sense channel [D0] - dc bus current on op-amp0
      {0x40,0xD0,0x00,0x00},
// end of sequence [40], current sense channel [D0] - dc bus current on op-amp0
      {0x40,0xD0,0x00,0x00},
// end of sequence [40], current sense channel [D0] - dc bus current on op-amp0
      {0xC0,0xD0,0x00,0x00},
// end of list + no int [C0], current sense channel [D0] - dc bus current on op-amp0
      {0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00}
  };

volatile unsigned short ADC0ResultList[2][RESULT_NO] =
{
    {32758, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0}
};
volatile unsigned short ADC1ResultList[2][RESULT_NO] =
{
    {32758, 32758, 32758, 32758, 0, 0, 0, 0},
    {32758, 32758, 32758, 32758, 0, 0, 0, 0}
};
PR_SECTION(DEFAULT_SEC)
```

5.1.6 adc.c: adc0_init()

Action

Before

```
void adc0_init(void)
{
      ADC0CTL_0_ACC_CFG = 3;        // Dual access mode
      ADC0CTL_0_STR_SEQA = 1;        // Store result at abort/restart
      ADC0CTL_1_CSL_BMOD = 0;        // Command list is single buffered
      ADC0CTL_1_RVL_BMOD = 1;        // Result list is double buffered

      ADC0TIM = 2;      // clock: clk = fbus [50 MHz]/(2x(reg.value + 1))
                //[0.25 - 8.33MHz]; 2 => 8.33 MHZ @ 50 MHz bus clock !_!

      ADC0FMT_DJM = 0;            // Left justified result data
      ADC0FMT_SRES = 4;           // 12-bit result

      // ADC0 Command Base Pointer
      ADC0CBP = ADC0CommandList;
      // ADC0 Result Base Pointer
      ADC0RBP = ADC0ResultList;

      // ADC0 Command/Result Offset registers
      ADC0CROFF1 = (unsigned char)(((unsigned long)&ADC0ResultList[1][0] -
               (unsigned long)&ADC0ResultList[0][0])>>1);

      ADC0CTL_0_ADC_EN = 1;     // enable ADC0
      ADC0EIE = 0xEE;           // enable all errors interrupts
}
```

After

```
void adc0_init(void)
{
      ADC0CTL_0_ACC_CFG = 3;        // Dual access mode
      ADC0CTL_0_STR_SEQA = 1;        // Store result at abort/restart
      //ADC0CTL_1_CSL_BMOD = 0;        // Command list is single buffered
      //ADC0CTL_1_RVL_BMOD = 1;        // Result list is double buffered

      ADC0TIM = 2;      // clock: clk = fbus [50 MHz]/(2x(reg.value + 1))
                //[0.25 - 8.33MHz]; 2 => 8.33 MHZ @ 50 MHz bus clock !_!

      ADC0FMT_DJM = 0;            // Left justified result data
      ADC0FMT_SRES = 4;           // 12-bit result

      // ADC0 Command Base Pointer
      ADC0CBP = ADC0CommandList;
      // ADC0 Result Base Pointer
      ADC0RBP = ADC0ResultList;

      // ADC0 Command/Result Offset registers
     ADC0CROFF1 = 0;

      ADC0CTL_0_ADC_EN = 1;     // enable ADC0
      ADC0EIE = 0xEE;           // enable all errors interrupts
}
```

5.1.7 adc.c: adc1_init()

	Action
Before	

```
void adc1_init(void)
{
  ADC1CTL_0_ACC_CFG = 3;      // Dual access mode
  ADC1CTL_0_STR_SEQA = 1;     // Store result at abort/restart

  ADC1TIM = 2;            // clock: clk = fbus [50 MHz] / (2x(reg.value + 1))
                 //[0.25 - 8.33MHz]; 2 => 8.33 MHZ @ 50 MHz bus clock !_!
  ADC1FMT_DJM = 0;            // left justified result data
  ADC1FMT_SRES = 4;           // 12-bit result

  ADC1CONIE_1_CON_IE1 = 0;      // End of sequence interrupt enable

  // ADC1 Command Base Pointer
  ADC1CBP = ADC1CommandList;
  // ADC1 Result Base Pointer
  ADC1RBP = ADC1ResultList;

  // ADC1 Command/Result Offset registers
  ADC1CROFF1 = 0;

  ADC1CTL_0_ADC_EN = 1;       // enable ADC1
  ADC1EIE = 0xEE;             // enable all error interrupts
}
```

Table continues on the next page...

Table continued from the previous page...

	Action

After	`void adc1_init(void)`
	`{`
	`  ADC1CTL_0_ACC_CFG = 3;      // Dual access mode`
	`  ADC1CTL_0_STR_SEQA = 1;     // Store result at abort/restart`
	`  ADC1CTL_1_CSL_BMOD = 0;       // Command list is single buffered`
	`  ADC1CTL_1_RVL_BMOD = 1;       // Result list is double buffered`
	
	`  ADC1TIM = 2;            // clock: clk = fbus [50 MHz] / (2x(reg.value + 1))`
	`                //[0.25 - 8.33MHz]; 2 => 8.33 MHZ @ 50 MHz bus clock !_!`
	`  ADC1FMT_DJM = 0;            // left justified result data`
	`  ADC1FMT_SRES = 4;           // 12-bit result`
	
	`  ADC1CONIE_1_CON_IE1 = 0;      // End of sequence interrupt enable`
	
	`  // ADC1 Command Base Pointer`
	`  ADC1CBP = ADC1CommandList;`
	`  // ADC1 Result Base Pointer`
	`  ADC1RBP = ADC1ResultList;`
	
	`  // ADC1 Command/Result Offset registers`
	` ADC1CROFF1 = (unsigned char)(((unsigned long)&ADC1ResultList[1][0]`
	`        - (unsigned long)&ADC1ResultList[0][0])>>1);`
	
	`  ADC1CTL_0_ADC_EN = 1;       // enable ADC1`
	`  ADC1EIE = 0xEE;             // enable all error interrupts`
	`}`

5.1.8 ptu.c: Definition

	Action

Befor	`PR_SECTION(ptuTrigE)`
	`volatile short ptuTriggerList0[PTU_LISTS_NO][PTU_COMMANDS] =`
	`{`
	`    {125,250,375,500,0x0000},`
	`    {200,400,600,800,0x0000}`
	`}; // !_! for 50 MHz bus clock`
	`volatile short ptuTriggerList1[PTU_LISTS_NO][PTU_COMMANDS] =`
	`{`
	`    {150,500,0x0000,0x0000,0x0000},`
	`    {150,500,0x0000,0x0000,0x0000}`
	`};`
	`PR_SECTION(DEFAULT_SEC`

Table continues on the next page...

Table continued from the previous page...

	Action

After	
	```
	PR_SECTION(ptuTrigE)
	volatile short ptuTriggerList0[PTU_LISTS_NO][PTU_COMMANDS] =
	{
	    {150,500,0x0000,0x0000,0x0000},
	    {150,500,0x0000,0x0000,0x0000}
	}; // !_! for 50 MHz bus clock
	volatile short ptuTriggerList1[PTU_LISTS_NO][PTU_COMMANDS] =
	{
	    {125,250,375,500,0x0000},
	    {200,400,600,800,0x0000}
	};
	PR_SECTION(DEFAULT_SEC)
	```

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support