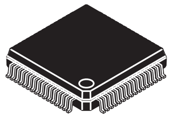
Freescale Semiconductor
Advance Information

Document Number: MC33816
Rev. 4.0, 4/2014

SD6 Programmable Solenoid Controller for Precision Solenoid Control Applications

33816

SOLENOID CONTROLLER



AE SUFFIX (PB-FREE)
98ASA00237D
64-PIN LQFP
EXPOSED PAD

The 33816 is a SMARTMOS programmable gate driver IC for precision solenoid control applications. The IC consists of five external MOSFET high-side pre-drivers and seven external MOSFET low-side pre-drivers. The 33816 provides a flexible solution for MOSFET's gate drive with a versatile control and optimized latency time. Gate drive, diagnosis, and protection are managed through four independent microcores, and two Code RAM and two Data RAM banks.

The IC contains two internal voltage regulators with overvoltage and undervoltage monitoring and protection. There are four current sense modules and VDS monitoring for fault detection and annunciation via a serial peripheral interface (SPI).

The device includes both individual charge pump outputs for each high-side pre-drivers and a high-voltage DC-DC converter low-side pre-driver.

These features along with cost effective packaging, make the 33816 ideal for powertrain engine control applications.

Features

- Battery voltage range, 5.0 V < V_{BATT} < 32 V
- Pre-drive operating voltage up to 72 V
- High-side/ low-side pre-drive PWM capability up to 100 KHz–30 nC
- All pre-drivers have four selectable slew rates
- Eight selectable, pre-defined VDS monitoring thresholds
- Encryption for microcode protection
- Integrated 1.0 MHz back-up clock

Applications

- Automotive (12 V), Truck and Industrial (24 V) Powertrain
- Diesel and Gasoline Direct Injection
- Transmission

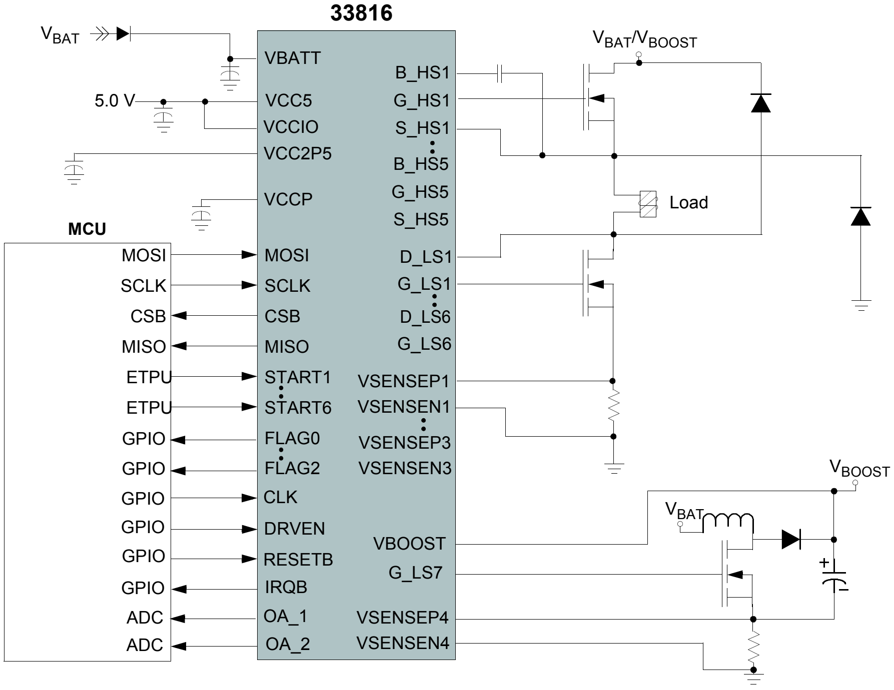


Figure 1. MC33816 Simplified Application Diagram

Table of Contents

1 Orderable Parts

This section describes the part numbers available to be purchased along with their differences. Valid orderable part numbers are provided on the web. To determine the orderable part numbers for this device, go to http://www.freescale.com and perform a part number search for the following device numbers.

Table 1. Orderable Part Variations

Part Number (1)	Temperature (T_A)	Package
MC33816AE	-40 to 125 °C	LQFP 64-pin exposed pad

Notes

1. To order parts in Tape & Reel, add the R2 suffix to the part number.

1.1 Cipher Key

Contact a Freescale sales representative to obtain devices with a specific encryption key and the associated code encryptor.

2 Internal Block Diagram

2.1 Simplified Internal Diagram

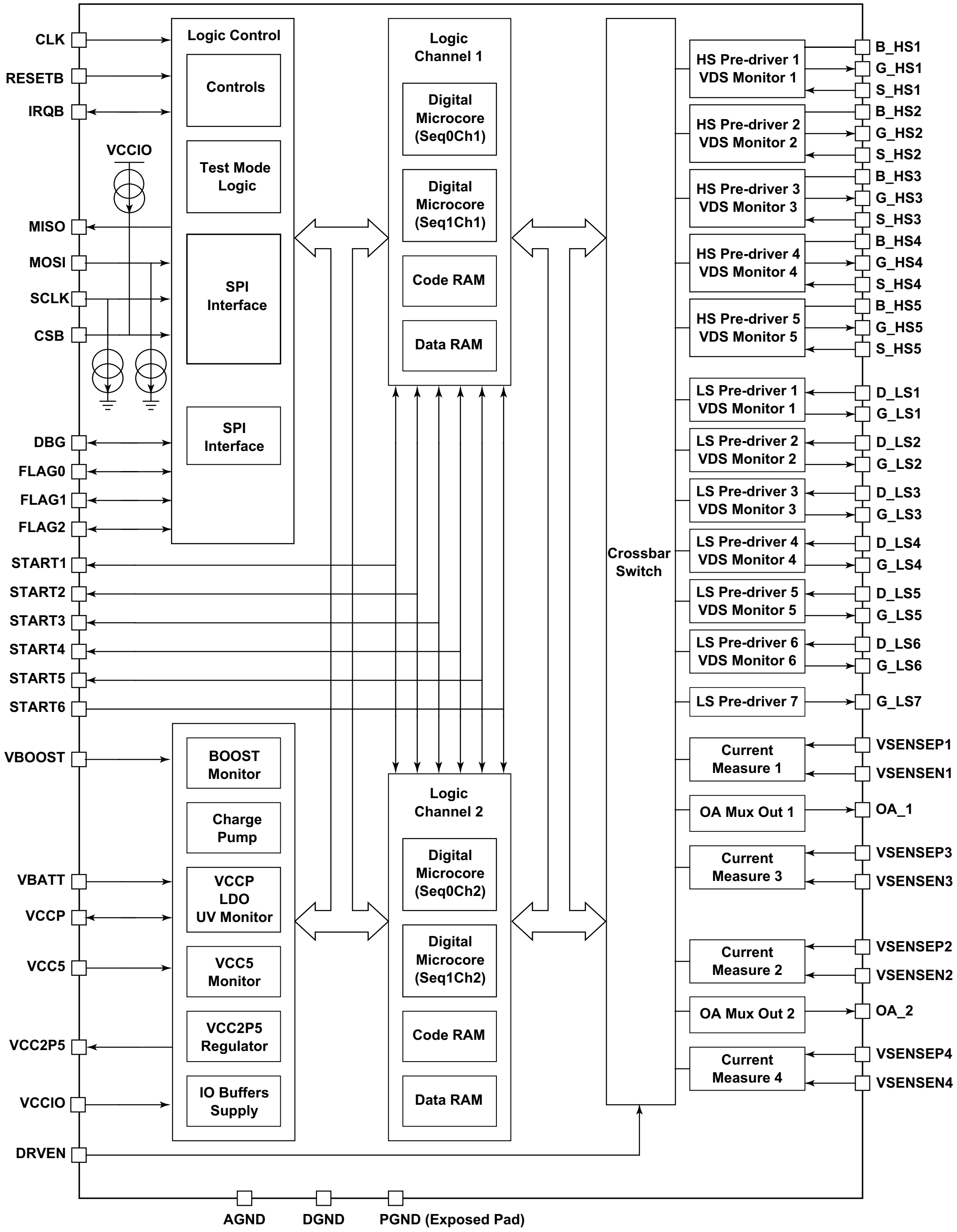


Figure 2. 33816 Simplified Internal Block Diagram

3 Pin Connections

3.1 Pinout Diagram

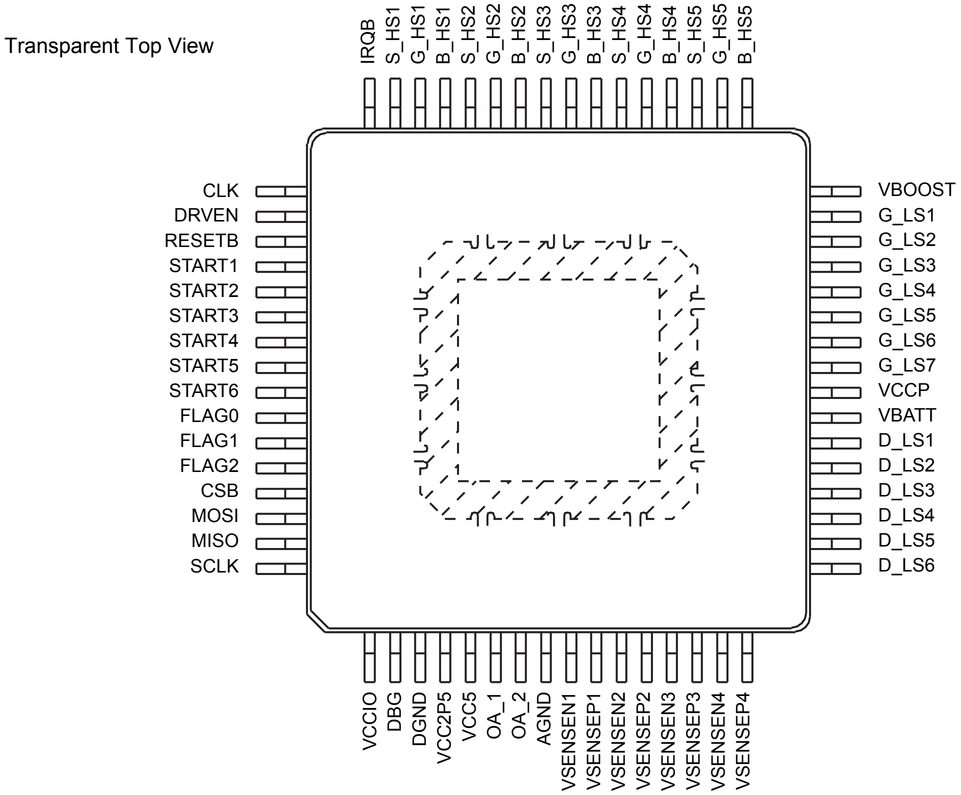


Figure 3. 33816 64-Pin LQFP-EP Pinout Diagram

3.2 Pin Definitions

Table 2. 33816 Pin Definitions

Pin Number	Pin Name	Pin Function	Definition
1	CLK	Input	External clock pin - internal weak pull-up (2)
2	DRVEN	Input	Driver enable pin - internal weak pull-down (4)
3	RESETB	Input	Device reset pin - internal weak pull-up (2)
4	START1	Input/Output	Trigger pin actuator 1/Flag_bus(3) - internal configurable pull-up/pull-down (3) (5)
5	START2	Input/Output	Trigger pin actuator 2/Flag_bus(4) - internal configurable pull-up/pull-down (3) (5)
6	START3	Input/Output	Trigger pin actuator 3/Flag_bus(5) - internal configurable pull-up/pull-down (3) (5)
7	START4	Input/Output	Trigger pin actuator 4/Flag_bus(6) - internal configurable pull-up/pull-down (3) (5)
8	START5	Input/Output	Trigger pin actuator 5/Flag_bus(7) - internal configurable pull-up/pull-down (3) (5)
9	START6	Input/Output	Trigger pin actuator 6/Flag_bus(8) - internal configurable pull-up/pull-down (3) (5)

Table 2. 33816 Pin Definitions (continued)

Pin Number	Pin Name	Pin Function	Definition
10	FLAG0	Input/Output	General purpose I/O/Flag_bus(0) - internal weak pull-down (4)
11	FLAG1	Input/Output	General purpose I/O/Flag_bus(1) - internal weak pull-down (4)
12	FLAG2	Input/Output	General purpose I/O/Flag_bus(2) - internal weak pull-down (4)
13	CSB	Input	SPI chip select - internal pull-up (3)
14	MOSI	Input	SPI slave data input - internal weak pull-up (2)
15	MISO	Output	SPI slave data output
16	SCLK	Input	SPI clock - internal weak pull-up (2)
17	VCCIO	Input	Digital I/O buffer supply (3.3 V or 5.0 V)
18	DBG	Input/Output	Debug pin/Flag_bus(12) - internal weak pull-up (2)
19	DGND	Ground	Digital ground
20	VCC2P5	Output	Internal 2.5 V digital power supply output/decoupling capacitor required
21	VCC5	Input	Power supply input pin (5.0 V)
22	OA_1	Output	Current sense analog output pin/Flag_bus(10) - internal weak pull-down (4)
23	OA_2	Output	Current sense analog output pin/Flag_bus(11) - internal weak pull-down (4)
24	AGND	Ground	Analog ground
25	VSENSEN1	Input	Current sense input comparator -
26	VSENSEP1	Input	Current sense input comparator +
27	VSENSEN2	Input	Current sense input comparator -
28	VSENSEP2	Input	Current sense input comparator +
29	VSENSEN3	Input	Current sense input comparator -
30	VSENSEP3	Input	Current sense input comparator +
31	VSENSEN4	Input	DC-DC current sense input comparator -
32	VSENSEP4	Input	DC-DC current sense input comparator +
33	D_LS6	Input	Low-side MOSFET drain pin monitor 6
34	D_LS5	Input	Low-side MOSFET drain pin monitor 5
35	D_LS4	Input	Low-side MOSFET drain pin monitor 4
36	D_LS3	Input	Low-side MOSFET drain pin monitor 3
37	D_LS2	Input	Low-side MOSFET drain pin monitor 2
38	D_LS1	Input	Low-side MOSFET drain pin monitor 1
39	VBATT	Input	Battery input voltage
40	VCCP	Input/Output	Internal 7.0 V power supply output pin/External 7.0 V power supply input pin
41	G_LS7	Output	DC-DC low-side MOSFET gate pin actuator 7
42	G_LS6	Output	Low-side MOSFET gate pin actuator 6
43	G_LS5	Output	Low-side MOSFET gate pin actuator 5
44	G_LS4	Output	Low-side MOSFET gate pin actuator 4
45	G_LS3	Output	Low-side MOSFET gate pin actuator 3
46	G_LS2	Output	Low-side MOSFET gate pin actuator 2
47	G_LS1	Output	Low-side MOSFET gate pin actuator 1

Table 2. 33816 Pin Definitions (continued)

Pin Number	Pin Name	Pin Function	Definition
48	VBOOST	Input	DC-DC feedback pin/Boost voltage monitor pin
49	B_HS5	-	High-side MOSFET bootstrap pin 5
50	G_HS5	Output	High-side MOSFET gate pin actuator 5
51	S_HS5	Input	High-side MOSFET source pin monitor 5
52	B_HS4	-	High-side MOSFET bootstrap pin 4
53	G_HS4	Output	High-side MOSFET gate pin actuator 4
54	S_HS4	Input	High-side MOSFET source pin monitor 4
55	B_HS3	-	High-side MOSFET bootstrap pin 3
56	G_HS3	Output	High-side MOSFET gate pin actuator 3
57	S_HS3	Input	High-side MOSFET source pin monitor 3
58	B_HS2	-	High-side MOSFET bootstrap pin 2
59	G_HS2	Output	High-side MOSFET gate pin actuator 2
60	S_HS2	Input	High-side MOSFET source pin monitor 2
61	B_HS1	-	High-side MOSFET bootstrap pin 1
62	G_HS1	Output	High-side MOSFET gate pin actuator 1
63	S_HS1	Input	High-side MOSFET source pin monitor 1
64	IRQB	Input/Output	Interrupt output/Flag_bus(9) - internal weak pull-down (4)
Exposed pad	PGND	Ground	Power ground

Notes

2. Internal weak pull-up to V_{CCIO} is typically 480 kΩ - Refer to the Internal Pull-up and Pull-down section.
3. Internal pull-up to V_{CCIO} is typically 120 kΩ - Refer to the Internal Pull-up and Pull-down section.
4. Internal weak pull-down to AGND is typically 480 kΩ - Refer to the Internal Pull-up and Pull-down section.
5. Internal pull-down to AGND is typically 120 kΩ - Refer to the Internal Pull-up and Pull-down section.

4 General Product Characteristics

4.1 Maximum Ratings

Table 3. Maximum Ratings

All voltages are with respect to the power ground (PGND), unless otherwise noted. Exceeding these ratings may cause a malfunction or permanent damage to the device.

Symbol	Description (Rating)	Min.	Max.	Unit	Notes
$V_{BOOSTMAX}$	VBOOST pin voltage range • Steady-state • Unpowered device	 0.0 –	 72 72	V	 (7)(9)
V_{BATT}	Battery voltage range (VBATT)	-0.3	72	V	(9)
V_{CC5}	VCC5 input pin	-0.3	18	V	
V_{CCIO}	VCCIO input pin	-0.3	18	V	
V_{CCP}	VCCP input/output pin	-0.3	9.0	V	
V_{CC2P5}	VCC2P5 output pin	-0.3	3.0	V	
V_{MAX_LOGIC}	SPI interface and logic input and output voltage (CSB, MOSI, MISO, SCLK, CLK, RESETB, IRQB, DRVEN, START1, START2, START3, START4, START5, START6, FLAG0, FLAG1, FLAG2, DBG, OA_1, OA_2)	-0.3	18	V	
V_{DGND}	Digital ground (DGND)	-0.3	0.3	V	
V_{AGND}	Analog ground (AGND)	-0.3	0.3	V	
V_{S_HSX}	Source high-side MOSFET pin (S_HS1, S_HS2, S_HS3, S_HS4, S_HS5) • Nominal • Transients t < 400 ns • Transients t < 800 ns • Unpowered device	 -3.0 -8.0 -6.0 –	 $V_{BOOSTMAX}$ $V_{BOOSTMAX}$ $V_{BOOSTMAX}$ 40	V	 (6) (6) (7)
V_{B_HSX}	Bootstrap high-side MOSFET pin (B_HS1, B_HS2, B_HS3, B_HS4, B_HS5) • Nominal • Transients t < 400 ns • Transients t < 800 ns • Unpowered device	 -0.3 -4.0 -2.0 –	 $V_{S\_HSX}+V_{BS\_HSX\_CL}$ $V_{S\_HSX}+V_{BS\_HSX\_CL}$ $V_{S\_HSX}+V_{BS\_HSX\_CL}$ $V_{S\_HSX}+V_{BS\_HSX\_CL}$	V	(9) (6) (6) (6)(7)
V_{G_HSX}	Gate high-side MOSFET pin (G_HS1, G_HS2, G_HS3, G_HS4, G_HS5)	V_{S_HSX} - 0.3	V_{B_HSX} + 0.3	V	(7)(8)
V_{G_LSX}	Gate high-side MOSFET pin (G_LS1, G_LS2, G_LS3, G_LS4, G_LS5, G_LS6, G_LS7) • Nominal • Transients t < 5.0 ns	 -0.3 -1.5	 V_{CCP} + 0.3 V_{CCP} + 1.5	V	 (6)(10)

Table 3. Maximum Ratings (continued)

All voltages are with respect to the power ground (PGND), unless otherwise noted. Exceeding these ratings may cause a malfunction or permanent damage to the device.

V_{D_LSX}	Drain high-side MOSFET pin (D_LS1, D_LS2, D_LS3, D_LS4, D_LS5, D_LS6) • Nominal • Transients t < 400 ns • Unpowered device	-3.0 -8.0 –	75 75 40	V	 (6) (6)(7)
V_{SENSEP}	Current measurement positive input pin voltage (VSENSEP1, VSENSEP2, VSENSEP3) • Static at VCC5 < 10 V • Dynamic for max 5.0 µs, 1.0 kHz repetition rate at VCC5 < 5.25 V • Dynamic for max 1.0 µs at VCC5 < 5.25 V	-2.5 -5.0 -15	2.5 5.0 15	V	 (6) (6)
V_{SENSEN}	Current measurement negative input pin voltage (VSENSEN1, VSENSEN2, VSENSEN3) • Static at VCC5 < 10 V • Dynamic for max 5.0 µs, 1.0 kHz repetition rate at VCC5 < 5.25 V • Dynamic for max 1.0 µs at VCC5 < 5.25 V	-1.0 -5.0 -15	1.0 5.0 15	V	 (6) (6)
$V_{SENSEP4}$	Current measurement four positive input pin voltage (VSENSEP4) • Static at VCC5 < 10 V • Dynamic for max 5.0 µs, 1.0 kHz repetition rate at VCC5 < 5.25 V • Dynamic for max 1.0 µs at VCC5 < 5.25 V	-4.2 -5.0 -15	2.5 5.0 15	V	 (6) (6)
$V_{SENSEN4}$	Current measurement four negative input pin voltage (VSENSEN4) • Static at VCC5 < 10 V • Dynamic for max 5.0 µs, 1.0 kHz repetition rate at VCC5 < 5.25 V • Dynamic for max 1.0 µs at VCC5 < 5.25 V	-3.0 -5.0 -15	1.0 5.0 15	V	 (6) (6)
  V_{ESD1-1} V_{ESD1-2} V_{ESD1-3} V_{ESD2-1} V_{ESD2-2}	ESD Voltage Human Body Model (HBM) • All pins • VBOOST, VBATT, S_HSx • D_LSx CDM • All pins • Corner pins (CLK, SCLK, VCCIO, VSENSEP4, D_LS6, VBOOST, B_HS5, IRQB)	 ±2000 ±4000 ±8000 ±500 ±750		V	(11)

Notes

6. This parameter is derived mainly from simulation.
7. In case of application power-off just after the power-down all the system capacitors connected the pins VBATT, VBOOST, VS_HSx, VG_HSx and VD_LSx are slowly discharged due to highly resistive discharge paths. A voltage remains on these pins until full capacitor discharge.
8. Relative voltage is referenced to the corresponding pre-driver channel biasing.
9. The differential voltage V_{BATT}-V_{B_HSx} must not exceed 40 V.
10. Considering V_{CCP} = 8.0 V - Energy of pulses < 0.0 V or > V_{CCP} limited to 2.0 µJ.
11. ESD testing is performed in accordance with the Human Body Model (HBM) (C_{ZAP} = 100 pF, R_{ZAP} = 1500 Ω), and the Charge Device Model (CDM), Robotic (C_{ZAP} = 4.0 pF).

4.2 Thermal Characteristics

Table 4. Thermal Ratings

Symbol	Description (Rating)	Min.	Typ.	Max.	Unit	Notes
 T_A T_J	Operating Temperature • Ambient • Junction	 -40 -40	 – –	 125 150	°C	
T_{STG}	Storage Temperature	-40	–	150	°C	
$R_{\Theta JA}$	Thermal Resistance • Junction-to-Ambient	–	27	–	°C/W	(12), (13)
T_{PPRT}	Peak Package Reflow Temperature During Reflow	–	–	Note 14	°C	(14)

Notes

12. Considering four layer FR4 PCB and 5.5 x 5.5 mm², with the exposed pad connected to the inner ground layer through 16 vias (Outer diameter: 0.3 mm, Inner diameter: 0.25 mm).
13. This parameter is derived mainly from simulation. (for the $R_{\theta JA}$ parameter)
14. Freescale’s Package Reflow capability meets Pb-free requirements for JEDEC standard J-STD-020C. For Peak Package Reflow Temperature and Moisture Sensitivity Levels (MSL), go to www.freescale.com, search by part number [e.g. remove prefixes/suffixes and enter the core ID to view all orderable parts. (i.e. MC33xxxD enter 33xxx), and review parametrics.]

4.3 Operating Conditions

This section describes the operating conditions of the device. Conditions apply to all the following data, unless otherwise noted.

Table 5. Operating Conditions

Characteristics noted under conditions -40 °C < T_A< +125 °C, referenced to DGND pin, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
V_{CC5}	VCC5 supply input voltage	4.75	5.0	5.25	V	
$V_{CC5\_DIGITAL}$	VCC5 supply input voltage for digital part functional only	4.0	5.0	5.25	V	(15)
V_{CCIO}	VCCIO supply input voltage	3.0	–	5.25	V	
V_{BATT}	VBATT power supply input voltage, Internal VCCP regulator, Normal operation	9.0	13.5	16	V	
V_{BATT_BR}	VBATT power supply input voltage, Internal VCCP regulator, Broken alternator regulator condition • Duration ≤ 1.0 hour	16	–	18	V	
V_{BATT_CRANK}	VBATT power supply input voltage, Internal or external VCCP regulator, Cranking condition	5.0	–	9.0	V	(16)
V_{BATT_JSTART}	VBATT power supply input voltage, Internal VCCP regulator, Jump start condition • T_A = 40 °C, Duration ≤ 2.0 min.	18	–	28	V	
$V_{BATT\_LOADDUMP}$	VBATT power supply input voltage, Internal VCCP regulator, Load dump • Duration ≤ 500 ms	18	–	40	V	
V_{BATT_EXT}	VBATT power supply input voltage, External VCCP regulator, Normal operation	9.0	–	32	V	(17)
$V_{BATT\_BR\_EXT}$	VBATT power supply input voltage, External VCCP regulator, Broken alternator regulator condition • Duration ≤ 1 hour	32	–	36	V	(17)
$V_{BATT\_JSTART\_EXT}$	VBATT power supply input voltage, External VCCP regulator, Jump start condition • T_A = 40 °C, Duration ≤ 15 min.	36	–	48	V	(17)
$V_{BATT\_LOADDUMP\_EXT}$	VBATT power supply input voltage, External VCCP regulator, Load dump • Duration ≤ 500 ms	36	–	58	V	(17), (18)
V_{BOOST}	Boost output voltage	5.0	–	$V_{BOOSTMAX}$	V	

Notes

15. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.
16. Full device functionality is guaranteed under cranking condition. However some derating can be observed on gate driver switching times and other parameters.
17. For 24 V system applications, the VCCP voltage must be externally supplied to limit power dissipation within the MC33816. Moreover, the MOSFETs' drain voltages must not exceed the high-side pre-driver pins max. ratings, even during transient conditions.
18. Implementation of a transient suppressor circuitry is highly recommended to avoid exceeding the max. rating.

4.4 Supply Currents

This section describes the current consumption characteristics of the device, as well as the conditions for the measurements.

All measurements are without output loads.

Table 6. Current Consumption Summary

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, referenced to DGND pin, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
I_{VCC5}	VCC5 supply current • f_{SYS} = 24 MHz, no microcore running • f_{SYS} = 24 MHz, all microcores running	– –	46 51	51 56	mA	(19)
I_{VCCIO}	VCCIO supply current • f_{SYS} = 24 MHz, no microcore running • f_{SYS} = 24 MHz, all microcores running	– –	45 1.0	70 –	µA mA	(19)
I_{VBATT_QUIESC}	VBATT power supply current in reset state V_{CC5} = V_{CCIO} = 0.0 V • V_{BATT} = 13.5 V • V_{BATT} = 40 V	– –	– –	180 800	µA	
I_{VBATT_OPER}	VBATT power supply current in normal operation V_{BATT} = 16 V • DRVEN low, internal VCCP reg. off • DRVEN low, Internal VCCP reg. on • DRVEN high, VCCP max load 65 mA	– – –	1.7 4.4 69.7	2.5 6.0 71	mA	
I_{VBOOST_QUIESC}	Boost supply current in reset state VCC5 = VCCIO = 5.0 V • V_{BOOST} = 13.5 V • V_{BOOST} = 40 V • V_{BOOST} = 65 V	40 150 250	– – –	65 280 450	µA	
I_{VBOOST_OPER}	Boost supply current in normal operation • V_{BOOST} = 16 V • V_{BOOST} = 48 V • V_{BOOST} = 65 V	– – –	4.2 4.55 4.9	4.85 5.35 5.75	mA	(20)

Notes

19. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.
20. The main current contributor is the charge pump, typically consuming 4.2 mA at VBOOST = 65 V

5 General Description

5.1 Introduction

The 33816 is a mixed signal IC for engine injector and electrical valve control, which provides a cost effective, flexible, and smart, high-side and low-side MOSFET gate drivers. The device includes both individual charge pump outputs for each high-side pre-driver and high-voltage DC-DC converter pre-driver. Gate drive, diagnosis, and protection against external faults, are managed through four independent and concurrent digital microcores using an extensive set of 93 microcode instructions. Each of the two logic channels, comprised of two microcores, has its own Code RAM and Data RAM. The internal microcode is protected against theft via encryption and corruption via check sums. All functions are designed to minimize the number of external components required.

5.2 Features

High-side and Low-side Pre-drivers

- Five high-side pre-drivers for logic level N-channel MOSFETs using four programmable slew rates
- Six low-side pre-drivers for logic level N-channel MOSFETs using four programmable slew rates
- Integrated bootstrap circuitry for each high-side pre-driver
- Integrated charge pump circuitry for each high-side pre-driver with 100% duty cycle capability

DC-DC Converter

- One low-side pre-driver, for a logic level N-channel MOSFET, can be optionally dedicated to providing a boost DC-DC converter with four programmable slew rates
- Boost voltage monitoring (with integrated feedback)

Current Measurement and Diagnostic

- Four independent current measurement blocks, including A/D converters with programmable gain, which are based on 8-bit D/A converters
- One current measurement (channel 4) is optionally configurable to support DC-DC converter with overload detection
- Five high-side and six low-side pre-drivers with independent VDS monitoring (eight programmable values) for fault protection and diagnostics
- Integrated load biasing to $V_{BATT}/2$ for diagnosis (on all high-side sources and all low-side drains)
- Capable of detecting missing ground connections

Power Supplies and Monitoring

- Integrated 7.0 V linear regulator (VCCP) for HS/LS power supply (optionally externally supplied for 24 V battery system), with undervoltage monitoring
- Integrated 2.5 V linear regulator (VCC2P5) for digital core supply based on VCC5 input supply, with undervoltage monitoring
- External 5.0 V supply (VCC5) with under/overvoltage monitoring
- Temperature monitoring
- Selectable VCCIO external supply (5.0 V or 3.3 V) for digital I/O

Digital Block

- Four digital microcores, each with their own ALU, and full access to the system crossbar switch
- Two memory banks: 1024 x 16-bit of code RAM with built-in error detection and 64 x 16-bit of data RAM
- A system-wide crossbar switch for analog resources configuration
- Memory BIST activated by the SPI, with pass/fail status

PLL and Backup Clock

- 12 to 24 MHz PLL internal system clock based on 1.0 MHz input clock
- Loss of clock protection by means of internal backup 1.0 MHz clock

Control Interface

- 16-bit slave SPI up to 10 MHz – two protocols – programmable slew rate
- 13 general purpose digital IOs – configurable through registers and microcode
- Direct pre-driver inhibition input
- Device reset input
- Hardware interrupt output

Miscellaneous

- Built-in encryption for microcode protection
- External digital I/O able to sustain voltages up to 18 V
- High ESD performance
- ESD strategy optimized for ESD System Level Stress (“System-efficient ESD Design”)
- High ESD holding voltage (>80 V)
- AEC-Q100 Rev G compliant
- Heavy duty compliant
- Enhanced analog testability based on JTAG

5.3 Block Diagram

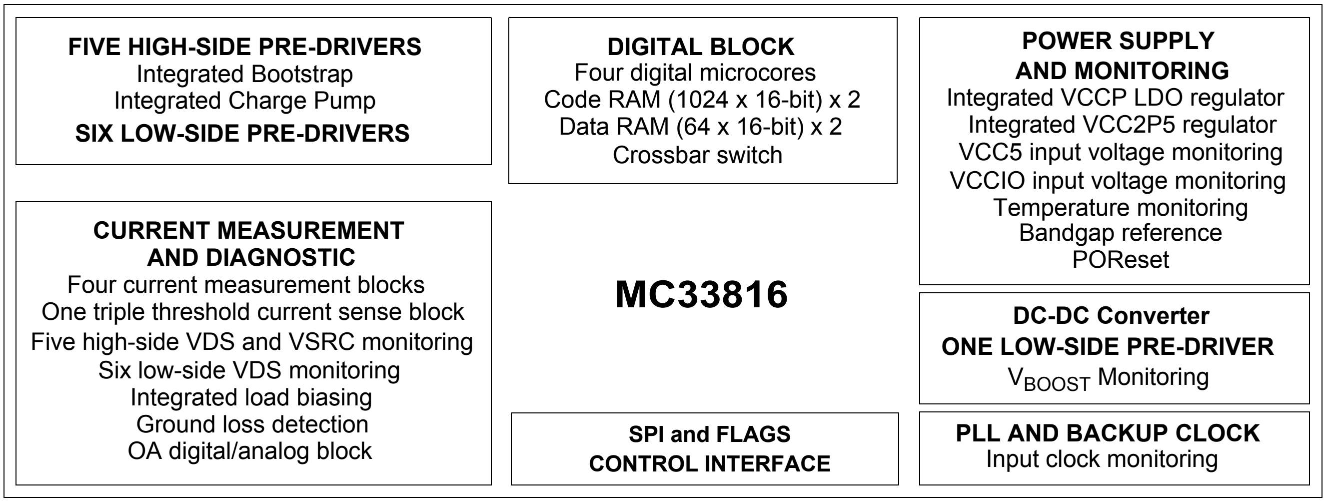


Figure 4. MC33816 - Functional Block Diagram

5.4 Functional Description

The general architecture consists of the combination of a set of four programmable microcores, integrated high-side and low-side pre-drivers for driving discrete power MOSFETs, measurement functions and means for diagnosis, and protection against external faults. Both battery voltage and booster voltage level high-side configurations are supported.

The chip communicates with the main controller through an SPI bus and a flexible set of direct interface signals.

The microcode managing the gate pre-drivers and diagnostics, is downloaded via the SPI. Data RAM and configuration registers are loaded via the SPI before or after the microcode download.

A 1.0 MHz clock signal is up-converted to an internal 24 MHz clock, by an internal PLL, to clock each of the four microcores on their own phase of a 6.0 MHz clock derived from the 24 MHz internal clock. The microcores are enabled by writing the suitable register (Flash_enable of channel 1 (0x100) and Flash_enable of channel 2 (0x120)).

The main MCU can reset the device at any time through the RESETB pin. The gate drivers are enabled by setting the drive enable signal applied on the DRVEN pin to a logic one.

The initial gate actuation sequence start is performed by bringing the appropriate STARTx input pin high.

Faults are reported to the MCU via the SPI or the Flag pins, if they are programmed as outputs. The IRQB pin can be used to interrupt the MCU when a fault occurs.

6 Functional Block Description

6.1 Power Up/Down Sequence

The recommended power up procedure to properly start up the MC33816 is shown in the following timing diagram.

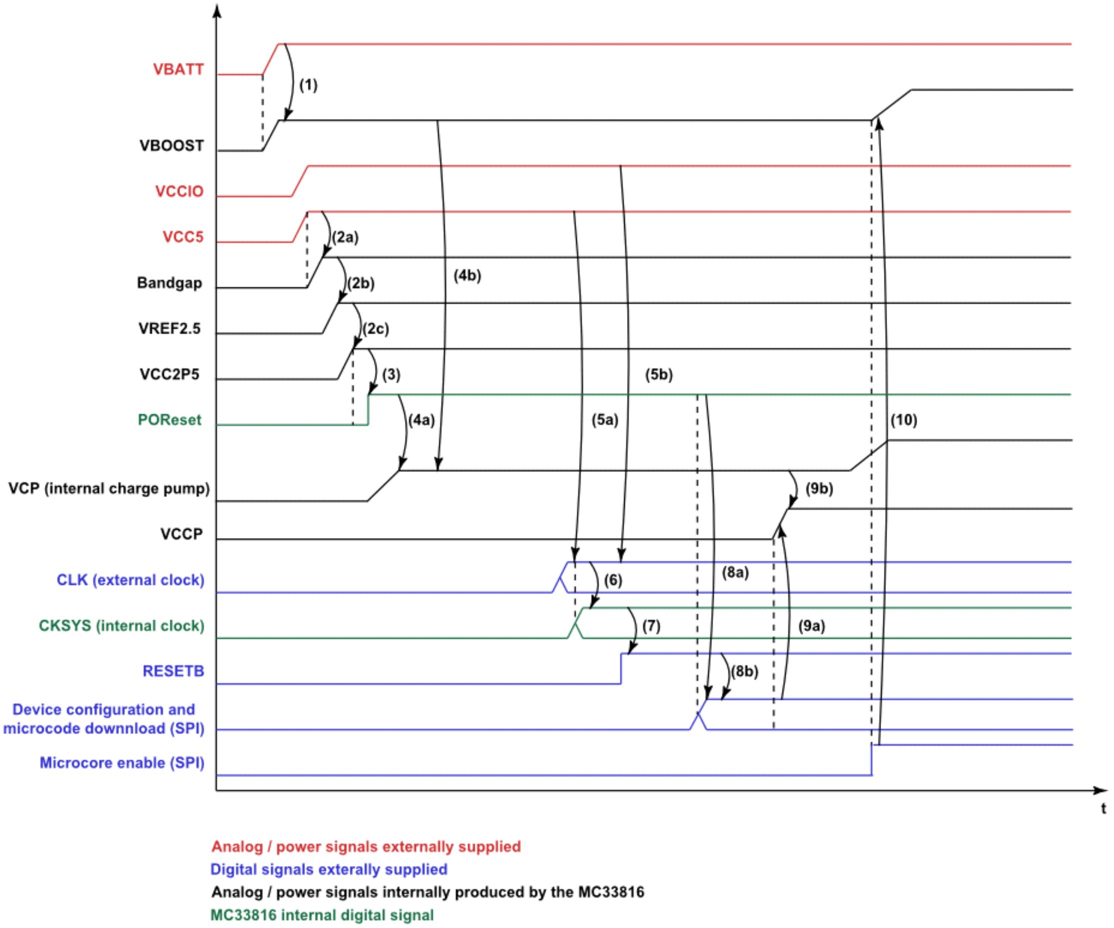


Figure 5. Power Up Sequence Timing Diagram

The detailed power up sequence description is provided below.

Table 7. Power Up Sequence Description

Phase	Sequence Description
(1)	Once a voltage is applied to VBATT, the voltage applied to VBOOST pin grows to V_{BATT}-V_D. V_D is the voltage drop across the diode of the boost external circuitry.
(2a)	Once a stable voltage is applied to the VCC5 pin the internal bandgap starts.
(2b)	Once the internal bandgap output is stable the VREF2.5 reference voltage regulator starts.
(2c)	Once the VREF2.5 reference voltage output is stable the VCC2P5 voltage regulator starts.
(3)	Since VCC2P5 output voltage is in its expected output voltage range the POReset is released.
(4a) (4b)	The internal charge pump starts when POReset is released and the suitable voltage is applied to the VBOOST pin.

Table 7. Power Up Sequence Description (continued)

(5a) (5b)	The external CLK signal or any digital signal (IO) is taken into account (input signals) or produced (output signals) by the MC33816 since VCCIO and VCC voltage are supplied.
(6)	Since a stable input signal at 1.0 MHz is applied to the CLK pin the internal PLL starts.
(7)	Since the internal PLL is stable and locked the main MCU can release the reset signal by setting the RESETB pin to the high state.
(8a) (8b)	The device configuration and microcode download through SPI communication can start once the VCC2P5 voltage is stable so the POReset is released and $t_{DIGIOREADY}$ time is reached. Moreover the RESETB pin states must be high.
(9a) (9b)	The internal V_{CCP} regulator is disabled by default and can then be enabled by SPI if no external V_{CCP} voltage is applied to the VCCP pin. Moreover the internal charge pump must be operational for allowing the internal regulator V_{CCP} to start.
(10)	The Microcore can be enabled and the BOOST DC-DC converter starts. The MC33816 is now ready to start load actuation accordingly to signal applied the STARTx pin.

During power up the voltage on VBATT pin can be higher than the voltage on the VBOOST pin.

The device is tolerant of various ramp-ups or slopes on the voltage supplies. There is no dependence on voltage sequencing of the power supplies. The only requirement is that the power supplies always remain below their maximum allowable values.

To power down the 33816 properly, it is recommended to assert the RESETB pin to the low state then switch off the V_{CC5}, V_{CCIO} and V_{BATT} external supplies while injection or actuation is not occurring.

A remaining voltage is present on the VBOOST pin until the boost output capacitor full discharge. This slow boost capacitor discharge must be considered with care to avoid any injury or system damage.

6.2 Power Supplies and Monitoring

The 33816 must be supplied by two external voltage sources, VBATT and VCC5. VCCIO must be connected to either a 5.0 V or 3.3 V source, depending on the logic levels desired.

The 33816 provides internal regulators to supply its own V_{CC2P5} and V_{CCP} voltages. V_{BOOST} can be generated via external circuitry connected to the LS7 pre-driver, and monitored by the current sense block 4 and the V_{BOOST} monitor input.

Table 8 provides an overview of the voltage supplies monitorings and capabilities.

Table 8. 33816 Power Supplies Overview

Power Supply Name	Purpose	Nominal Voltage	Nominal Current	Externally Supplied or Internally Generated	Source of Power
VCC5	Powers VCC2P5	5.0 V	51 mA	Externally	External regulator
VCCIO	Digital I/O buffer supply	3.3 or 5.0 V	1.0 mA	Externally	External regulator
VBATT	Provides V_{BAT} voltage and generates V_{CCP} voltage (if not provided externally) V_{CC5} and V_{CCIO} must be provided externally	V_{BAT} - 0.7 V	4 to 70 mA	Externally	Vehicle battery with reverse battery protection
VBOOST	Power for injector actuation	V_{BOOST}	5.0 mA (21)	Externally	Boost converter or V_{BAT}
Bandgap	Internal reference	1.3 V	–	Internally	V_{CC5}
VREF2.5	1.0% reference for DACs	2.5 V	–	Internally	V_{CC5}
VCC2P5	Supply for logic core	2.5 V	15 mA	Internally	V_{CC5}
VCCP	Gate voltage supply for low-side and high-side pre-drivers	7.0 V	65 mA max.	Internally or externally	V_{BAT} or external regulator

Table 8. 33816 Power Supplies Overview (continued)

VCP (charge pump)	Gate drive for high-side switches in case of bootstrap circuitry unavailable	V_{BOOST} + 8.0 V max.	375 µA min.	Internally	V_{BOOST} + V_{CC5}

Note

21. MC33816 internal consumption.

6.2.1 Band Gap Reference

In order to achieve the precision required, the device contains a 1.27 V band gap voltage reference. This band gap reference is accurate to ±2.0% over the full temperature range. The band gap input is supplied by the external 5.0 V supply.

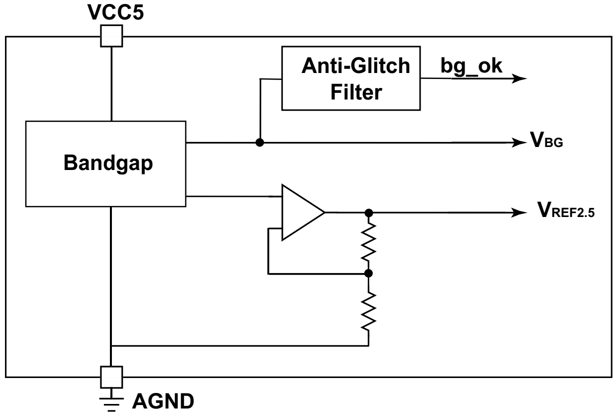


Figure 6. Bandgap Reference Overview

During power up, as soon as V_{CC5} is above V_{CC5_BGMIN}, the band gap reference is started. When the band gap voltage is stable at the target level, and after a delay time of $t_{BG\_OK\_AGF}$ generated by an anti-glitch filter, the bg_ok signal is asserted. This signal is used to switch on the V_{CC2P5} regulator and to enable the V_{CCP} internal regulator.

At power down, the band gap reference is switched off at the V_{CC5} voltage switch off.

A second internal 2.5 V reference voltage $V_{REF2.5}$ is used by all the DACs. The reference voltage has a precision of ±1.0%.

Table 9. Band Gap Reference Electrical Specifications

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to AGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$V_{REF2.5}$	2.5 V reference voltage for DACs	2.475	2.5	2.525	V	
V_{CC5_BGMIN}	Minimum VCC5 voltage for Bandgap operating	–	–	3.8	V	
$t_{BG\_OK\_AGF}$	bg_ok anti-glitch filter time.	–	9.0	–	µs	(22)

Note

22. This parameter is derived mainly from simulation.

6.2.2 VCC2P5 and Power On Reset (POR)

The integrated V_{CC2P5} voltage regulator provides 2.5 V to supply the logic core of the device. A voltage monitor on the regulator output provides a Power On Reset to keep the logic reset until the V_{CC2P5} voltage is within the working range.

The V_{CC2P5} regulator input voltage is provided by the external V_{CC5} voltage input pin. The bg_ok signal must be asserted to allow the V_{CC2P5} regulator to start.

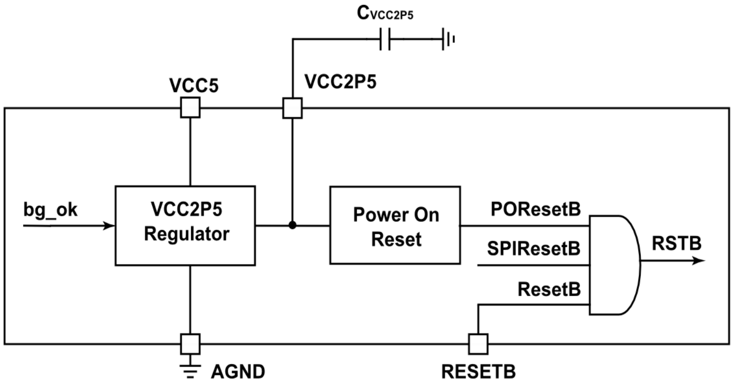


Figure 7. V_{CC2P5} Voltage Regulator and Power On Reset Overview

If the V_{CC2P5} voltage is below the undervoltage lockout threshold $V_{PORESETB-}$ for a minimum duration of $t_{PORESETB}$, the power on reset signal (POResetB) is asserted to the logic core after a delay of $t_{D\_PORESETB}$.

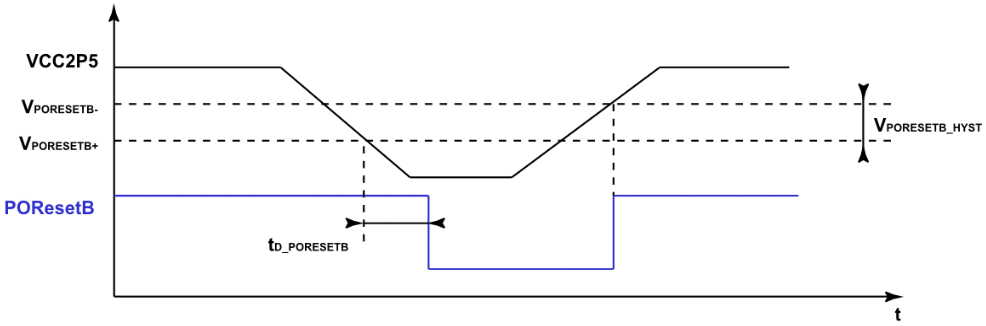


Figure 8. POResetB Diagram

The POResetB signal combines with the external reset signal ResetB, issued from the RESETB pin and the SPIResetB signal coming from the SPI interface. The AND gate output RSTB is used to reset the logic core and all device internal modules.

As long as RSTB is asserted, the SPI module is also inactive. The MCU can detect the reset state:

- either by sending any message to the device and checking for the control pattern on MISO during command word. In case of RSTB asserted the returned value is different from '0xA8'.
- or by reading out any register with a reset value not equal to zero (example: Device Identification register (0x1D5)). In case of RSTB asserted the returned value is '0x00'.

The logic core should be properly supplied with 2.5 V when 5.0 V is present at VCC5 pin (thus allowing logic core operations and communication with the microcontroller), even when no voltage is provided at the VBATT pin, and consequently no voltage is present on VCCP pin.

Table 10. VCC2P5 AND POR Electrical Specifications

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to AGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
V_{CC2P5}	VCC2P5 supply output voltage	2.375	2.5	2.625	V	(23)
I_{VCC2P5}	V_{CC2P5} supply output current f_{SYS} = 24 MHz, all microcores running	–	-15	-25	mA	(24)
I_{VCC2P5_LIM}	V_{CC2P5} supply output current limit	-50	-70	-90	mA	
ΔV_{VCC5}	V_{CC5} - V_{CC2P5} voltage dropout V_{CC5} = 4.0 V and I_{VCC2P5} = -25 mA	–	–	1.7	V	
$V_{PORESETB-}$	V_{CC2P5} voltage threshold for asserting POResetB	2.0	2.11	2.21	V	
$V_{PORESETB+}$	V_{CC2P5} voltage threshold for deasserting PORsetB	2.07	2.19	2.3	V	
$V_{PORESETB\_HYST}$	$V_{PORESETB}$ voltage hysteresis	50	75	100	mV	
$t_{D\_PORESETB}$	Time from undervoltage detection to POResetB assertion	–	0.7	1.5	µs	(24)
$t_{PORESETB}$	POResetB duration min. C_{VCC2P5}, min. $V_{PORESETB\_HYST}$ and max I_{VCC2P5_LIM}	361	–	–	ns	(24)

Note

23. Considering an external output capacitor C_{VCC2P5} minimum value of 0.5 µF, typically value of 1.0 µF, and maximum value of 3.0 µF.
24. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.2.3 VCC5 Voltage Monitoring

The VCC5 voltage is externally powered and internally monitored. It supplies the internal VCC2P5 regulator.

6.2.3.1 VCC5 Overvoltage Monitoring

If the voltage applied to the VCC5 pin exceeds the V_{OVVCC5} threshold, the device disconnects after the T_{D_OVVCC5} delay, the VCC5 pin from the circuitry it powers, until the voltage returns to normal. This feature protects the VCC5 pin during a short to battery, up to a maximum voltage of 18 V.

A VCC5 pin voltage above the V_{OVVCC5_VCCP} threshold shuts down the VCCP internal regulator until the VCC5 voltage returns to its normal value.

Table 11. VCC5 Overvoltage Monitoring Electrical Specifications

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to AGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
V_{OVVCC5}	VCC5 overvoltage threshold	7.5	8.5	10.0	V	
t_{D_OVVCC5}	VCC5 overvoltage switch time • Differential input voltage = 1.0 V	–	–	1.0	µs	(25)
V_{OVVCC5_VCCP}	VCC5 overvoltage threshold for VCCP shutdown	6.2	6.9	7.5	V	

Note

25. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.2.3.2 VCC5 Undervoltage Monitoring

VCC5 undervoltage monitoring is used to disable all the pre-drivers, whenever the supply voltage at the VCC5 pin is not high enough to guarantee full functionality of the analog modules of the device. The output signal uv_vcc5 of this undervoltage monitoring is routed to all the pre-drivers and combined with uv_vccp signal. In the digital core, the uv_vcc5 is set high in the Driver_status (0x1D2) register when a VCC5 undervoltage condition is detected. In addition, an interrupt request (if a suitable interrupt vector is enabled in the Driver_config register (0x1C5)) is issued to the microcontroller, as soon as uv_vcc5 is asserted.

Table 12. VCC5 Undervoltage Monitoring Electrical Specifications

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to AGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$V_{UVVCC5-}$	VCC5 undervoltage low-voltage threshold	4.3	4.45	4.7	V	
$V_{UVVCC5+}$	VCC5 undervoltage high-voltage threshold	4.35	4.5	4.75	V	
V_{UVVCC5_HYST}	VCC5 undervoltage hysteresis	30	50	85	mV	
t_{D_UVVCC5}	VCC5 undervoltage switching time • Differential input voltage = 1.0 V	–	–	150	ns	(26)
t_{FILTER_UVVCC5}	VCC5 undervoltage anti-glitch filter delay time	0.8	1.3	2.0	µs	(26)

Note

26. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.2.4 VCCP LDO Regulator

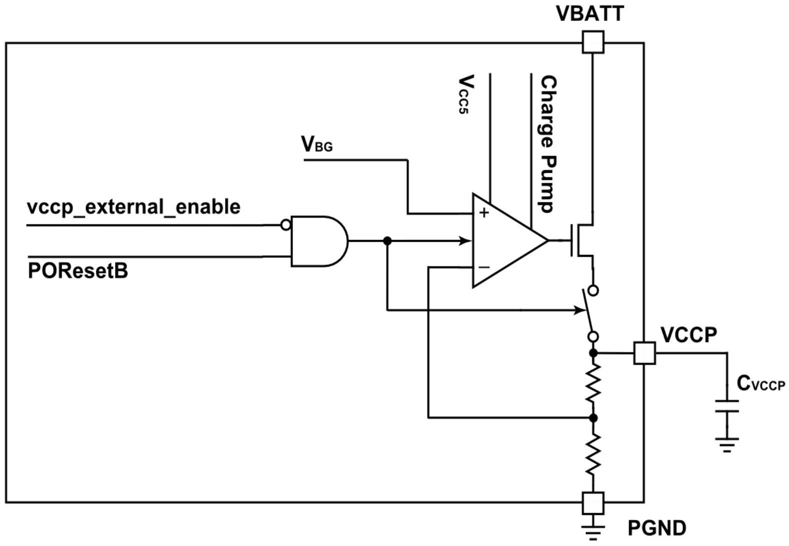


Figure 9. VCCP LDO Regulator

The voltage source at the VBATT input pin provides power for the VCCP regulator. This integrated linear regulator provides typically 7.0 V at the VCCP pin, to supply the pre-driver section of the device. The regulator uses low drop out features to extend the system's operating range when V_{BATT} temporarily falls below its normal operating range, for example during engine crank conditions. This avoids problems caused by insufficient gate voltage, such as slow MOSFET switching and increased on-state losses. A capacitor is required at the VCCP pin to provide the high peak currents required when charging a MOSFET gate.

The low dropout mode of the regulator is active only when the voltage at VCC5 is above the VCC5 undervoltage threshold $V_{UVVCC5+}$. At low V_{CC5}, the regulator may be active, but with an increased dropout voltage.

At power-up, the VCCP regulator is activated only when the band gap voltage is stable at its nominal value and, therefore, the POResetB signal is released high. When the voltage at VBATT exceeds its undervoltage lockout threshold at typically 4.7 V, the internal charge pump becomes active and enables the VCCP regulator.

If V_{CC5} is not present or low, POResetB is active and the VCCP regulator is disabled.

The VCCP node can also be powered by an external voltage source connected to the VCCP pin. This external source is recommended for 24 V applications. The internal VCCP regulator is sized for 12 V system operation, including the ISO voltage transients specified for those systems. But for 24 V system operation, the internal VCCP linear regulator dissipates too much power. In this case, the internal VCCP regulator should be switched off via the vccp_external_enable signal, by setting the vccp_ext_en bit of the Driver_status register (0x1C5) to ‘1’, and using an external supply.

Using an external regulator introduces the possibility of the VCCP voltage being greater than the battery voltage and potentially sourcing current from VCCP to VBATT. The internal regulator’s back-to-back MOSFETs avoid this problem by blocking such current when the regulator is disabled.

The VCCP regulator is controlled via the vccp_external_enable signal from the digital core. The VCCP regulator is switched off by default after reset. When the VCCP regulator is disabled during a reset condition (i.e. RESETB pin is low), or when the vccp_external_enable signal is high, it is switched off to reduce quiescent current drawn from the VBATT pin.

Table 13. V_{CCP} LDO Regulator Electrical Specifications

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
V_{CCP}	V_{CCP} output voltage • 0.0 mA ≤ I_{VCCP} ≤ 65 mA	6.5	7.0	7.5	V	(27)
V_{CCP_EXT}	V_{CCP} input voltage range (V_{CCP} externally supplied)	5.0	–	9.0	V	
I_{VCCP}	V_{CCP} output current (average during PWM operation) • 9.0 V < V_{BATT} < 18 V	–	–	-65	mA	(28)
I_{VCCP_MAX}	V_{CCP} output current limitation	-100	-150	-200	mA	
ΔV_{VCCP}	V_{BATT} to V_{CCP} voltage dropout • V_{BATT} = 5.0 V and I_{VCCP} = - 65 mA • V_{BATT} = 5.0 V and I_{VCCP} = - 50 mA • V_{BATT} = 5.0 V and I_{VCCP} = - 30 mA • V_{BATT} = 5.0 V and I_{VCCP} = - 10 mA	 – – – –	 – – – –	 350 280 170 60	mV	

Note

27. Considering an external output capacitor C_{VCCP} connected to PGND pin with a minimum value of 1.0 μF, a typically value of 4.7 μF, and a maximum value of 14 μF.
28. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.2.4.1 VCCP Undervoltage Monitoring

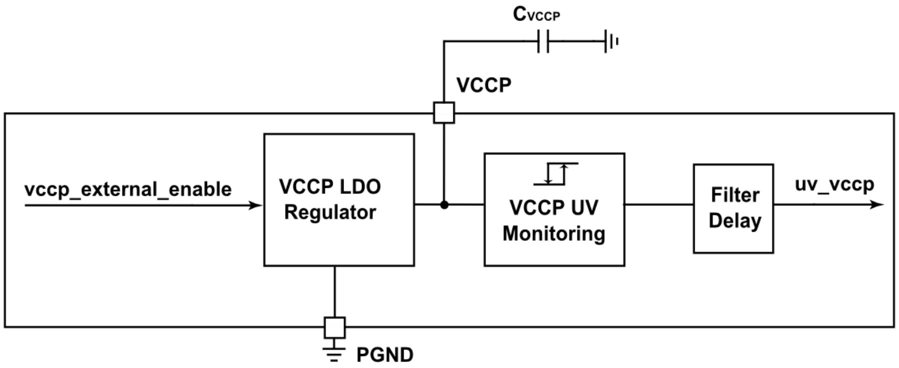


Figure 10. VCCP Undervoltage Monitoring

Whether an internal or external (24 V applications) VCCP regulator is implemented, the V_{CCP} voltage is internally monitored by a voltage comparator to detect voltages below the minimum operating range. When V_{CCP} falls below its undervoltage threshold $V_{UVVCCP-}$, the gate driver outputs are automatically switched off by the digital core.

The gate drivers are re-enabled after the V_{CCP} voltage rises above the $V_{UVVCCP+}$ upper threshold, and after a t_{FILTER_UVVCCP} filter delay.

When an undervoltage occurs, operations are stopped before insufficient gate driver supply voltage causes a malfunction.

Moreover, during a battery voltage disconnection, VCCP quickly decays, causing all MOSFETs to be switched off before the VBATT external input capacitor completely discharges. The digital core monitors the undervoltage comparator output (uv_vccp) to implement the protection strategies described previously. In addition, if the vccp_irq_en bit of the Driver_status register (0x1C5) is set high, an interrupt request is issued to the microcontroller through the IRQB pin as soon as uv_vccp signal is asserted and the uv_vccp flag bit in the Driver_status register (0x1D2) is set high.

Table 14. V_{CCP} Undervoltage Monitoring Electrical Specifications

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to AGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$V_{UVVCCP-}$	VCCP undervoltage low-voltage threshold	4.30	4.50	4.68	V	
$V_{UVVCCP+}$	VCCP undervoltage high-voltage threshold	4.40	4.55	4.73	V	
V_{UVVCCP_HYST}	VCCP undervoltage hysteresis	30	50	70	mV	
t_{D_UVVCCP}	VCCP UV switching time • 1.0 V differential input voltage.	–	–	2.5	μs	(29)
t_{FILTER_UVVCCP}	VCCP UV anti-glitch filter delay time	0.8	1.3	2.0	μs	(29)

Note

29. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.2.5 DC-DC Converter

The boost converter uses low-side pre-driver LS7, current measurement block four, external passive components, and the VBAT supply to create an output voltage up to 72 V. Figure 11 shows one of two possible topologies that differ in how the boost capacitor is connected. A more detailed block diagram of current measurement block four (Figure 21) shows it has two positive and one negative current comparators.

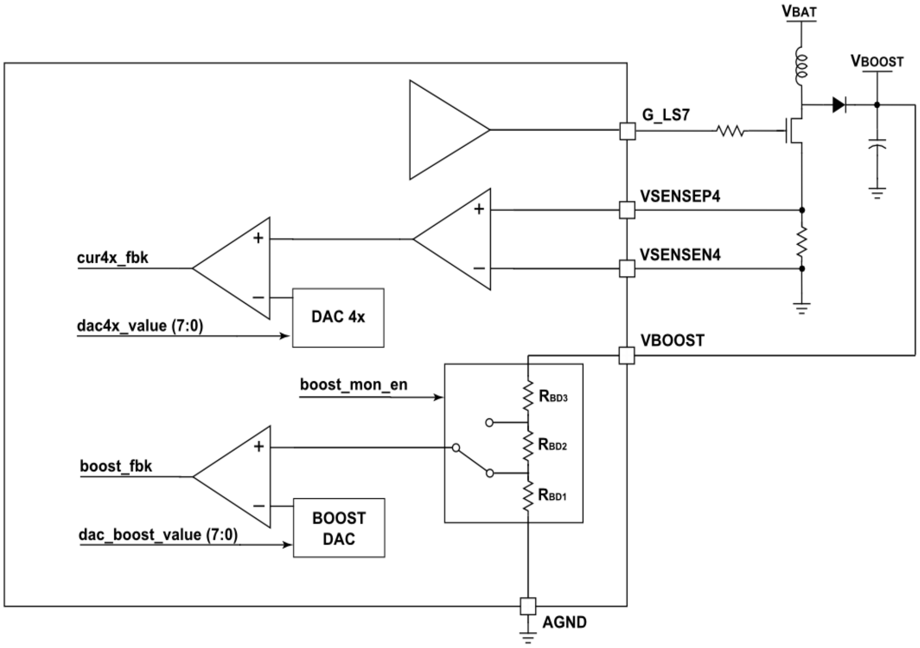


Figure 11. Boost Converter Topology for Fixed Frequency Mode

The key principle driving a boost converter is the tendency of an inductor to resist changes in current. When the switch is turned on, the current flows through the inductor and energy is stored in it. When the switch is turned off, the inductor transfers all or part of its stored energy into its output capacitance and the load. The inductor's voltage polarity changes such that it adds to the input voltage. Thus, the voltage across the inductor and the input voltage are in series, and together charge the output capacitor through the diode to a voltage greater than the input voltage.

The boost converter requires a V_{BATT} voltage greater than 4.7 V to operate, and the device must not be in a reset state. A V_{CCP} voltage greater than the $V_{UVVCCP+}$ threshold enables the low-side driver. Boost operation can be inhibited by the DRVEN pin (low state), if the ls7_ovr bit of the Driver_config register (0x1C5) is set to '0' (reset value is '1').

The boost voltage regulation loop is controlled by one of the microcores. The boost output voltage is set according to the boost_threshold, an 8-bit word in the Boost_dac register (0x19B). The boost comparator filter time and type can be specified in the Boost_filter register (0x19D).

The current measurement block four monitors the current through the low-side switch. Its two positive comparators allow asynchronous current regulation between the thresholds defined by the DAC4n_value (4:0) and the DAC4h_value (4:0). The negative comparator that uses the DAC4neg_value (4:0) allows diagnostics during injection by detecting over current drawn from the boost output capacitor in variable frequency mode only.

The boost regulator operates in one of two hysteretic modes: 'Variable Frequency' and 'Fixed Frequency'.

Both modes operate in a hysteretic mode based on the instantaneous voltage at VBOOST pin. The boost regulator turns on or off as the output voltage falls below or rises above a threshold window centered on the desired output. When the regulator is on, the 'Variable Frequency' and 'Fixed Frequency' modes control the power switch differently as described by the following.

6.2.5.1 Variable Frequency Mode

The variable frequency mode requires the topology shown in Figure 12. Note that in this topology the boost capacitor and the inductor share the current sense resistor, and the inductor current is accurately measured only when there is no load current in the output filter capacitor. Therefore, to ensure the inductor current never exceeds its saturation levels, the boost converter operation must be suspended during boost injection phases.

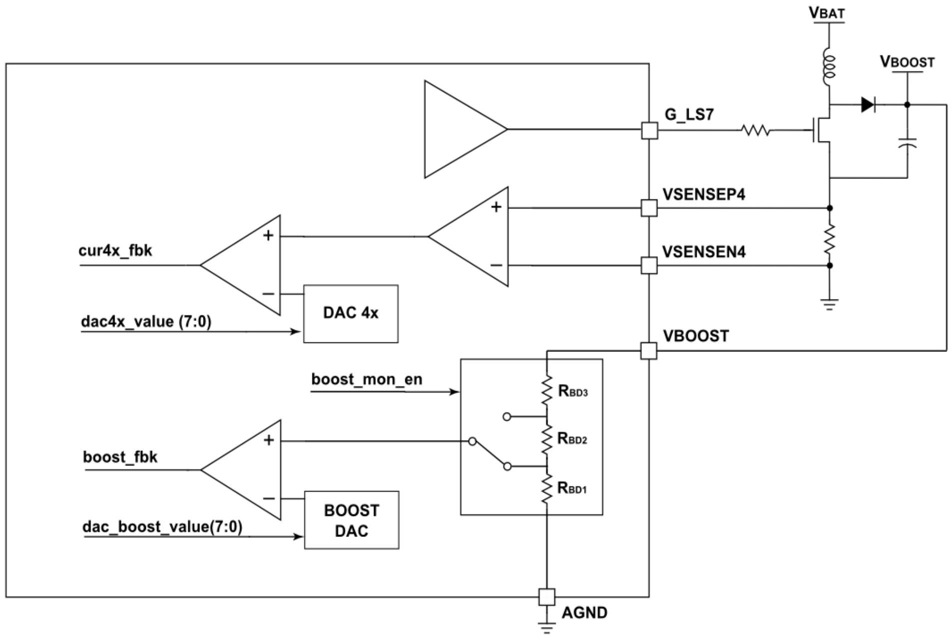


Figure 12. Boost Converter Topology for Variable Frequency Mode

In variable frequency mode, on/off switching is triggered by sense current falling below a lower current threshold and by rising above an upper current threshold. This mode uses a hysteretic current control loop within a hysteretic voltage control loop. Once the current thresholds are programmed, hardware controls the current regulation loop while microcode controls the voltage regulation loop. Duty cycle and frequency vary with operating conditions. While in the hysteretic current control mode, the converter operates 'asynchronously' because microcode does not directly control the timing of each switching edge. A dedicated internal circuitry enabled by microcode manages the current modulation.

Whenever V_{BOOST} falls below the desired regulation window, the boost circuit can be activated. The microcode must first set the upper V_{BOOST} threshold (boost_threshold) in the Boost_dac register (0x19B). The regulator must wait until the output of the boost comparator is valid, that is, after the vboost_filter has expired. Once the switching begins, energy is delivered to the output each cycle, the output voltage rises until it eventually exceeds the upper boost_threshold. At this time LS7 is turned off, the microcode sets boost_threshold to its low value, and the regulator pauses until the vboost_filter expires again. It then waits for V_{BOOST} to cross the lower threshold before beginning a new boost cycle.

6.2.5.2 Fixed Frequency Mode

The fixed frequency mode uses the topology shown in Figure 11. Because the VBOOST capacitor does not share the sense resistor with the MOSFET in this topology, it is not necessary to suspend the boost conversion during boost injection phases.

In fixed frequency mode, the microcode directly controls the period each switching cycle. The low-side switch is turned on and then off for a fixed time. All switch timing is under control of the microcode. A hysteretic voltage control loop starts and stops on/off cycling. Each time the boost is activated the microcode must first set the upper V_{BOOST} threshold and wait until the VBOOST comparator has settled before activating the power switch. When the V_{BOOST} reaches the upper threshold, switching is suspended and the V_{BOOST} threshold is reset to its lower level.

6.2.5.3 Boost Start Up Sequence

After V_{CCP} stabilizes above its $V_{UVVCCP+}$ undervoltage threshold and ResetB is released (i.e. RESETB pin is high), the microcode is launched by writing the pre_flash_enable bit of the suitable Flash_enable registers (0x100, 0x120) by SPI. The boost regulation starts immediately, unless the DRVEN pin controls LS7. In this case, the DRVEN must be set to high to start regulation.

Using a software based soft start routine is highly recommended. This is accomplished by incrementing the boost_threshold in the Boost_dac register (0x19B) in the microcode.

6.2.5.4 Low-side Pre-driver for DC-DC Converter (LS7)

The 33816 provides a seventh independent low-side pre-driver designed to drive the gate of external low-side configuration N-channel logic level MOSFET. This pre-driver dedicated to DC-DC conversion supports highest PWM frequency and can be used for general purpose.

The pre-driver does not have a diagnosis feature.

Internal to the device, a gate to source pull-down resistor holds the external MOSFETs in off state while the device is in a power on reset state (RSTB low).

This low-side pre-driver is supplied by V_{CCP} voltage.

The logic command ls7_command to switch the external MOSFET is provided by the digital microcore. This command is generated taking into account following signals:

- The signal DrvEn issued from the DRVEN pin is added to the control signal for the driver. As long as the DrvEn signal is negated, the low-side pre-driver is switched off. The low-side pre-driver for the DC-DC converter includes a feature to override the switch off path via a DrvEn signal. As long as the ls7_en_ovr bit of the Driver_config register (0x1C5) is set to ‘1’, the pre-driver is not influenced by the DrvEn signal.
- The V_{CCP} undervoltage signals (uv_vccp) issue from the V_{CCP} monitoring. During an undervoltage, the external MOSFET is switched off
- The V_{CC5} undervoltage signals (uv_vcc5) issued from the V_{CC5} monitoring. During an undervoltage, the external MOSFET is switched off
- The signal cksys_drven issued from the clock monitoring: In cases of a missing clock, the external MOSFET is switched off while the digital core has not switched to the internal backup clock. This condition can be optionally disabled by setting the bit cksys_missing_disable_driver of the Backup_clock_status_handle (0x1C7) register to ‘0’.
- The logic command coming from channel logic (ls7_in)

A truth table describing the status of the ls7_command signal is given in Table 15.

Table 15. Low-side Seven Pre-driver Truth Table

DrvEn	ls7_en_ovr	uv_vccp	uv_vcc5	cksys_drven	ls7_in	ls7_command	Driver Status
0	0	–	–	–	–	0(30)	off
–	–	1	–	–	–	0(30)	off
–	–	–	1	–	–	0(30)	off
–	–	–	–	0	–	0(30)	off
–	–	–	—	–	0	0(30)	off
–	1	0	0	0	1	1(31)	on
1	–	0	0	0	1	1(31)	on

Note

30. When ls7_command is low, the G_LS7 pin is driven low (pull-down to PGND voltage)
31. When ls7_command is high, the G_LS7 pin is driven high (pull-up to V_{CCP} voltage)

Table 16. Low-side Pre-drivers ON State Electrical Specifications

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
V_{G_LS7}	G_LS7 operating voltage	0.0	–	V_{CCP}	V	(32)
$f_{G\_LS7\_PWM}$	PWM frequency • 5.0 V ≤ V_{BATT} ≤ 18 V	0.0	–	300	kHz	(32)(33)(34)
DC_{G_LS7}	Duty cycle	0.0	–	100	%	(32)(33)

Table 16. Low-side Pre-drivers ON State Electrical Specifications (continued)

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$I_{G\_LS7\_PWM}$	G_LS7 current (average during PWM operation) • $Q_G = Q_{G\_LS7}$; f_{PWM} = 300 kHz • $Q_G = Q_{G\_LS7}$; f_{PWM} = 100 kHz • $Q_G = Q_{G\_LS7}$; f_{PWM} = 50 kHz	 – – –	 9.0 3.0 1.5	 22.5 7.5 3.75	mA	(32)
$I_{G\_LS7\_SRC}$	Peak source gate drive current	–	680	–	mA	(32)
$I_{G\_LS7\_SRC}$	Peak sink gate drive current at fastest slew rate setting with minimum R_{G_LS7} of 2.0 Ω and V_{CCP}/V_{GS} = 7.0 V	–	2200	–	mA	(32)
$t_{R\_G\_LS7}$	Turn on rise time at 1500 V/µs slew rate; 10%-90% of out voltage; V_{CCP} = 7.0 V; at Open pin	3.5	–	11	ns	(32)
$t_{F\_G\_LS7}$	Turn on fall time at 1500 V/µs slew rate; 10%-90% of out voltage; V_{CCP} = 7.0 V; at Open pin	3.5	–	11	ns	(32)
$t_{R\_G\_LS7}$	Turn on rise time at 300-25 V/µs slew rate; 10%-90% of out voltage; V_{CCP} = 7.0 V; at Open pin	5.0	–	25	ns	(32)
$t_{F\_G\_LS7}$	Turn on fall time at 300-25 V/µs slew rate; 10%-90% of out voltage; V_{CCP} = 7.0 V; at Open pin	5.0	–	25	ns	(32)
$t_{DON\_G\_LS7}$	Turn on propagation delay at 1500 V/µs slew rate; 10% of out voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	10	–	50	ns	(32)
$t_{DOFF\_G\_LS7}$	Turn off propagation delay at 1500 V/µs slew rate; 10% of out voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	10	–	50	ns	(32)
$t_{DON\_G\_LS7}$	Turn on propagation delay at 300 V/µs slew rate; 10% of out voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	10	–	70	ns	(32)
$t_{DOFF\_G\_LS7}$	Turn off propagation delay at 300 V/µs slew rate; 10% of out voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	10	–	70	ns	(32)
$t_{DON\_G\_LS7}$	Turn on propagation delay at 50 V/µs slew rate; 10% of out voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	15	–	100	ns	(32)
$t_{DOFF\_G\_LS7}$	Turn off propagation delay at 50 V/µs slew rate; 10% of out voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	15	–	100	ns	(32)
$t_{DOFF\_G\_LS7}$	Turn on propagation delay at 25 V/µs slew rate; 10% of out voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	15	–	120	ns	(32)
$t_{DOFF\_G\_LS7}$	Turn off propagation delay at 25 V/µs slew rate; 10% of out voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	15	–	120	ns	(32)

Note

32. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.
33. A series resistor to the MOSFET gate of 2.0 Ω must be implemented if using the fastest slew rate setting. For all the other slew rate settings the minimum resistor is 0 Ω.
34. The external low-side MOSFET gate charge must not exceed 75 nC. A gate charge of maximum 100 nC is admitted if the $f_{PWM} \leq$ 225 kHz.

Table 17. Low-side Pre-drivers OFF State Electrical Specifications

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
R_{PD_LS7}	G_LS7 to PGND pull-down resistor	25	50	90	kΩ	

6.2.5.4.1 Low-side Pre-driver Slew Rate Control

The driver strength can be selected among a set of four values by the SPI registers. The strength for the rising and falling edge can be chosen independently by the signals ls7_slewrate_p (1:0) and ls7_slew rate_n (1:0), issued by the digital core and accessible by means of the bits slew rate_ls7_rising(1:0) and slew rate_ls7_falling(1:0) in the Ls_slew rate register (0x18F).

The slew rate is determined by the PMOS and NMOS R_{DSON} of the push/pull driver circuitry.

The typical gate slew rate values are defined in Table 18 and Table 19. These values are given as reference and are impacted by the external circuitry.

Table 18. Low-side Seven Pre-drivers PMOS Slew Rate Settings

ls7_slewrate_p(1:0)	Slew Rate (V/µs)	RDSON_PMOS (switching ON) (Ohm)
00	1500	5.0
01	300	14.6
10	50	85
11	25	170

Table 19. Low-side Seven Pre-drivers NMOS Slew Rate Settings

ls7_slewrate_n(1:0)	Slew Rate (V/µs)	RDSON_NMOS (switching OFF) (Ohm)
00	1500	1.1
01	300	5.9
10	50	35
11	25	69

Table 20. Low-side Seven Pre-drivers Slew Rates Characteristics

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$R_{DS\_HSx\_p}$ (00)	G_HSx pMOS R_{DS_ON} (00), 1500 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 2.5 V • at external V_{GS} = 1.0 to 4.0 V;	 3.0 2.6	 5.0 –	 8.6 10.7	Ω	
$R_{DS\_HSx\_n}$ (00)	G_HSx nMOS R_{DS_ON} (00), 1500 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 2.5 V • at external V_{GS} = 1.0 to 4.0 V;	 0.6 0.5	 1.1 –	 2.0 2.9	Ω	
$R_{DS\_HSx\_p}$ (01)	G_HSx pMOS R_{DS_ON} (01), 300 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 2.5 V • at external V_{GS} = 1.0 to 4.0 V;	 8.8 7.5	 14.6 –	 25.3 31.3	Ω	
$R_{DS\_HSx\_n}$ (01)	G_HSx nMOS R_{DS_ON} (01), 300 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 2.5 V • at external V_{GS} = 1.0 to 4.0 V;	 3.4 2.5	 5.9 –	 11.1 16.5	Ω	
$R_{DS\_HSx\_p}$ (10)	G_HSx pMOS R_{DS_ON} (10), 50 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 1.0 to 4.0 V;	 61	 85	 115	Ω	
$R_{DS\_HSx\_n}$ (10)	G_HSx nMOS R_{DS_ON} (10), 50 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 1.0 to 4.0 V;	 23	 35	 50	Ω	

Table 20. Low-side Seven Pre-drivers Slew Rates Characteristics

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$R_{DS\_HSx\_p}$ (11)	G_HSx pMOS R_{DS_ON} (11), 25 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 1.0 to 4.0 V;	122	170	230	Ω	
$R_{DS\_HSx\_n}$ (11)	G_HSx nMOS R_{DS_ON} (11), 25 V/µs, V_{CCP} = 7.0 V at external V_{GS} = 1.0 to 4.0 V;	47	69	100	Ω	
t_{SLR_HS}	Slew rate switching time • 1 ck cycle at 6.0 MHz	–	–	166	ns	(35)

Note

35. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.2.5.4.2 Safe State of DC-DC Low-side Pre-driver

To guarantee a safe condition, the G_LS7 output is immediately forced to a low level, switching off the external MOSFET when reset is asserted, and the device is not operating. In this phase, the pre-driver is powered by the charge already stored in the V_{CCP} buffer capacitor. A low level output is guaranteed as long as a typical voltage greater than 1.1 V is available.

When the V_{CCP} supply voltage is lower than 1.1 V, the pre-driver output is pulled to PGND by an internal high resistance R_{PD_LS7} pull-down resistor.

6.2.5.5 Current Measurement for DC-DC Converter

The 4th current sense block is dedicated to the DC-DC convertor with a low-side current measurement, including a double positive threshold comparator and concurrently provide an overcurrent supervision at the booster capacitor.

The two-point current control of a DC-DC converter is optimized, such as to reach a low latency of the control loop. This architecture is able to provide a short delay from the VSENSEPx and VSENSENx inputs to the G_LS7 output.

The digital core contains hard wired logic for a two-point current regulation, using the cur4h_fbk and cur4l_fbk signals as inputs that directly drives the G_LS7 pin. Refer to the Current Measurement section for the detailed description and parameters.

A third comparator is implemented to detect negative current into the R_{SENSE} sense resistor. Refer to the Current Measurement for DC-DC Conversion section for the detailed description and parameters.

6.2.5.6 Boost Voltage Monitoring

The Boost voltage monitoring block is dedicated:

- to the V_{BOOST} voltage measurement, if the V_{BOOST} voltage is externally supplied, and when the block are in boost monitor mode
- or a Battery undervoltage measurement in UV V_{BOOST} mode when the VBOOST is connected to the device supplied (battery).

Table 21. Boost Voltage Monitoring Electrical Characteristics

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$V_{BOOSTMAX}$	Input voltage range	0.0	–	72	V	(36)
R_{VBOOST_IN}	Input impedance	400	640	–	kΩ	
G_{VBOOST_DIV}	V_{BOOST} voltage divider ratio (boost monitor mode)	1/32 *0.996	1/32	1/32 *1.004		
$G_{UV\_VBOOST\_DIV}$	V_{BOOST} voltage divider ratio (UV Vboost mode)	1/4 *0.996	1/4	1/4 *1.004		

Table 21. Boost Voltage Monitoring Electrical Characteristics

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$f_{CVBOOST\_DIV}$	V_{BOOST} analog filter cutoff frequency (boost monitor mode only)	50	100	200	kHz	
V_{VBOOST_REF}	DAC reference voltage	2.475	2.5	2.525	V	
$V_{VBOOST\_DAC\_LSB}$	DAC LSB	–	9.77	–	mV	(36)
$V_{VBOOST\_DAC\_OUT\_MIN}$	DAC minimum output voltage • DAC code = 0x00	–	0.0	–	V	(36)
$V_{VBOOST\_DAC\_OUT\_MAX}$	DAC maximum output voltage • DAC code = 0xFF	–	2.49	–	V	(36)
E_{VBOOST_DAC}	Total DAC error	–	–	0.2	%	
$V_{VBOOST\_DIV\_OFFSET}$	Total DAC including comparator offset	-20	–	20	mV	
V_{VBOOST_HYST}	Comparator hysteresis referred to V_{BOOST} (boost monitor mode)	112	160	208	mV	
V_{UV}_VBOOST_HYST	Comparator hysteresis referred to V_{BOOST} (UV Vboost mode)	10	20	30	mV	
t_{VBOOST_COMP}	Comparator switching time, Propagation delay + rise/fall time • 50 mV differential input voltage	–	–	1.0	µs	(36)
E_{VBOOST}	V_{BOOST} measurement total error • V_{BOOST} = 40 V and divider ratio 1/32 • V_{BOOST} = 4.85 V and divider ratio 1/4	 -2.0 -2.0	 – –	 2.0 2.0	%	
t_{VBOOST_DAC}	V_{BOOST} DAC settling time	–	–	0.9	µs	(36)

Note

36. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.2.6 Boost Voltage Monitoring Mode

Boost voltage monitoring mode is mostly used in Direct Injection (DI) applications when boost voltage is required to drive the injectors. The boost voltage monitor checks by means of a voltage comparator with a very accurate threshold eight bit DAC, regardless of whether V_{BOOST} exceeds the target value. The digital microcore acquires the comparator output for the boost voltage control and management.

An internal voltage divider network (R_{BD1}, R_{BD2}, and R_{BD3}) scales the boost voltage to be safely handled by the module. The operating voltage range on the VBOOST pin is up to V_{BOOST} max. A typical ratio of 1/32 is used for boost voltage monitoring.

The output signal of the voltage divider is filtered by an analog RC filter with a cutoff frequency of typically $f_{CVBOOST\_DIV}$, only for the ratio 1/32.

The hysteresis voltage comparator is supplied by VCC5 and referenced to AGND. If the boost voltage at the V_{BOOST_DIV} signal is above the DAC threshold, the comparator output boost_fbk is asserted, while it is set low if the V_{BOOST_DIV} voltage is below the DAC threshold. The comparator output boost_fbk is connected to the digital microcore.

The reference voltage DAC gets its unsigned input value from the signal dac_boost_value (7:0), issued from the digital cores. The boost voltage threshold can be calculated using the following formula:

- $V_{Boost} = DAC\ \_Value * V_{VBOOST\_DAC\_LSB}/G_{VBOOST\_DIV}$

DAC _Value is the decimal value of the DAC (dac_boost_value(7:0)).

$V_{VBOOST\_DAC\_LSB}$ is the DAC resolution.

G_{VBOOST_DIV} is the V_{BOOST} voltage divider ratio in boost monitor mode.

Due to the compensation concept, values below 0x08 must not be used. Also, values higher than 0xE1 must not be used, because this would result in a boost voltage exceeding the input voltage range $V_{BOOSTMAX}$. Practically, the boost voltage set point threshold must be set significantly below the $V_{BOOSTMAX}$, due to dynamic effects such as a voltage drop in the boost capacitor. DAC value clamping to 0xD0 is highly recommended.

Table 22. Boost Voltage DAC Value Examples in Boost Monitor Mode

DAC Value (hex)	DAC Value (dec)	DAC Output Voltage (mV)	V_{BOOST} Upper Threshold (V)		
			Minimum	Typical	Maximum
08	8	78	2.45	2.5	2.55
9A	154	1504	47.16	48.13	49.09
B0	176	1719	53.90	55.00	56.10
D0	208	2031	63.70	65.00	66.30
E1	225	2197	68.91	70.31	71.72

6.2.7 V_{BOOST} UV Monitoring Mode

In applications without boost voltage, the battery voltage is connected to the VBOOST pin to supply the internal charge pump. In such applications, the boost voltage monitoring can be used to detect an undervoltage at the VBOOST pin. For this purpose, the internal voltage divider ratio can be changed from 1/32 to 1/4, by setting the signal boost_mon_en to ’1’.

The V_{BOOST} UV monitor has the following characteristics different from the boost monitor mode:

- In this mode, the usable DAC range is limited from 0x08 to 0xF8 due to digital trimming.
- In this mode, the comparator output signal boost_fbk should be used in the digital core to disable all the high-side pre-drivers.This shut-off path is enabled by the signal vboost_disable_en inside the digital core. The uv_vboost signal goes high as soon as the voltage at the VBOOST pin is below the threshold, if the V_{BOOST} UV monitor is enabled (uv_vboost_en=1).

The digital filter used for the V_{BOOST} voltage measurement is activated for the V_{BOOST} UV monitoring mode. The DAC set point value in this mode has to be chosen, considering the pre-drivers must not be disabled for a battery voltage above 5.0 V, and the device internal charge pump works properly down to a battery voltage of $V_{UVVCC5+}$. This leads to a DAC set point value of 0x7C and the following values for UV V_{BOOST}:

- Undervoltage lower threshold (min.): 4.72 V
- Undervoltage upper threshold (max.): 4.94 V

The output signal uv_vboost (active high) of this undervoltage monitor is routed to all the high-side pre-drivers and combined with uv_vccp and uv_vcc5 signal to disable the pre-drivers. In the digital core, the bit uv_vboost the Driver_status register (0x1D2) is set when a V_{BOOST} undervoltage event occurs. In addition, an interrupt request is issued to the microcontroller as soon as uv_vboost is asserted, if the bit vboost_irq_en of the Driver_config register (0x1C5) is set to ’1’. The V_{BOOST} UV threshold can be calculated using the following formula.

- V_{BOOST} = DAC _Value * $V_{VBOOST\_DAC\_LSB}/G_{UV\_VBOOST\_DIV}$

DAC _Value is the decimal value of the DAC.

$V_{VBOOST\_DAC\_LSB}$ is the DAC resolution.

$G_{UV\_VBOOST\_DIV}$ is the V_{BOOST} voltage divider ratio in UV V_{BOOST} mode.

Table 23. Boost Voltage DAC Value Examples in UV V_{BOOST} Mode

DAC Value (hex)	DAC Value (dec)	DAC Output Voltage (mV)	V_{BOOST} UV Lower Threshold (V)	V_{BOOST} UV Upper Threshold (V)
08	8	78	0.28	0.32
7C	124	1213	4.72	4.94
89	137	1341	5.22	5.46
96	150	1468	5.72	5.98
F8	248	2427	9.47	9.88

6.2.8 Ground Disconnection

The device integrates three separate ground pins: PGND, DGND, and AGND:

- PGND is the substrate connection and it is only connected to the package exposed pad, to guarantee a low-impedance connection and get optimized EMC performances. PGND is the reference ground for the V_{CCP} regulator, some analog functions, and all of the low-side pre-drivers. It is highly recommended to directly connect PGND to the ECU ground plane.
- DGND is the reference ground for the digital logic core. It is highly recommended to directly connect DGND to the ECU ground plane.The microcontroller, as well as other logic devices communicating with the device should share the same reference ground connected to the ground plane to prevent noise.
- AGND is the ground for all the noise sensitive analog blocks integrated into the device, such as the bandgap reference, the current sense circuitry, and the output amplifiers (OA_x pins). This pin should be connected to the analog ground of the ECU. A star connection is recommended to guarantee a clean analog signal acquisition of the OAX_x pins from the MCU.

Due to their functionality, some analog functions are referred to PGND:

- V_{DS} monitors of the low-side drivers
- V_{SRC} monitors of the high-side drivers
- The load biasing S_HSX regulator and the D_LSx pull-down

All the ground pins of the device should be connected to the same ground voltage. Even during transient conditions, the voltage difference between PGND, DGND, and AGND must be limited to ±0.3 V. The layout of the ground connection of the ECU should be carefully designed, to limit the ground noise generated as much as possible, for instance during fast switching of the external power MOSFETs.

The decoupling and filter capacitors at the different supply voltage pins should be implemented as described by the following:

- VCC5 to AGND
- VCCIO to DGND
- VCC2P5 to DGND
- VCCP to PGND
- VBATT to PGND
- VBOOST to AGND or PGND

Table 24. Decoupling and Filter Capacitors Specification

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
C_{VCC5_AGND}	Capacitor connected between the VCC5 pin and AGND	–	100	–	nF	
C_{VCCIO_DGND}	Capacitor connected between the VCCIO pin and DGND	–	100	–	nF	
C_{VCC2P5_DGND}	Capacitor connected between the VCC2P5 pin and DGND	–	–	–	nF	
C_{VCCP_PGND}	Capacitor connected between the VCCP pin and PGND	–	–	–	nF	
C_{BATT_PGND}	Capacitor connected between the VBATT pin and PGND	–	–	–	nF	
$C_{VBOOST\_AGN\_PGND}$	Capacitor connected between the VBOOST pin and AGND or PGND	–	–	–	nF	

6.2.9 Detection of Missing GND Connections

The 33816 can detect any single or multiple missing connection of any ground pin (PGND, DGND, AGND) of the device.

At least one ground must remain connected to allow the loss of ground detection.

If the ground disconnection is detected, the internal signal uv_vccp is asserted and all the pre-drivers are disabled. The ground lost detection is filtered to allow the device to work in a proper way for a time of typically $t_{MISS\_GND\_DCT\_FLT}$ via the uv_vccp signal.

Table 25. Missing Ground Detection Specifications

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$t_{MISS\_GND\_DCT\_FLT}$	Missing ground detection filter time for uv_vccp asserted	–	1.3	–	ms	(37)

Note

37. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.2.10 Temperature Monitoring

The device includes a junction temperature monitoring feature, which monitors the junction temperature. If the maximum junction temperature is exceeded, the signal overtemperature going to the digital core is set. This signal over_temp is mapped to the overtemperature bit of the Driver_status Register (0x1D2) and can be monitored by the MCU. In addition, an interrupt request (if enabled in the Driver _config register (0x1C5)) is issued to the microcontroller as soon as over_temp signal is asserted.

Table 26. Temperature Monitoring Specifications

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$T_{THRESHOLD}$	Temperature monitoring threshold	167	177	187	°C	

6.2.11 Shut OFF Path Via DRVEN and 18 V Robustness

If the ECU detects a fault condition, it can disable all of the 33816’s output drivers via the driver enable pin DRVEN.

If the DRVEN input pin is low, all pre-drivers are switched off, but the digital core is fully functional. Access to the status of the DRVEN pin can be read via the SPI by reading the Driver_status register (0x1D2).

Most of the logic pins and the analog output pins are designed to be tolerant of a short to 18 V. Specifically, the following digital interface pins of the device are self-protected against a voltage of up to 18 V: CLK, IRQB, RESETB, DRVEN, MISO, MOSI, SCLK, CSB, DBG, STARTx (6x), FLAGx (3x), and OA_x (2x). The DRVEN function works even if these pins or the supply pins increase to 18 V.

To protect the device from an overvoltage up to 18 V at the VCC5 supply pin, there is an overvoltage detect circuit implemented on this pin. This function leads to the whole device being switched off during an overvoltage condition. Under this condition all pre-drivers are switched off.

Two switches protect the output structure of the digital I/O pins. One switch blocks current into the VCCIO supply path, while the other blocks current into the digital I/O pin for each digital I/O, to clamp the voltage below 10 V.

Table 27. Shut OFF Path Specification

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$t_{DIGIOREADY}$	Digital output ready time after POResetB deactivation	–	–	300	µs	(38) (39)

Note

38. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.
39. Adding a SPI response checking routine is highly recommended, to check the SPI functionality after $t_{DIGIOREADY}$ time has elapsed.

6.3 High-side Pre-drivers

The 33816 provides five independent high-side pre-drivers designed to drive the gate of external high-side configuration N-channel logic level MOSFETs. These pre-drivers are dedicated to load driving like injectors or solenoids, and integrate diagnosis features.

Internal to the device is a gate to source pull-down resistor holding the external MOSFETs in off state while the device is in a power on reset state (RSTB low).

The external FET can be connected to either V_{BATT} or a higher voltage V_{BOOST}.

The high-side pre-drivers are supplied by an external bootstrap capacitor connected between the S_HSX and B_HSX pins.

The driver slew rate can be selected individually for each of the five drivers, among a set of four value pairs by the SPI registers. All five drivers have identical electrical characteristics.

Any high-side pre-driver can also be used as a low-side pre-driver.

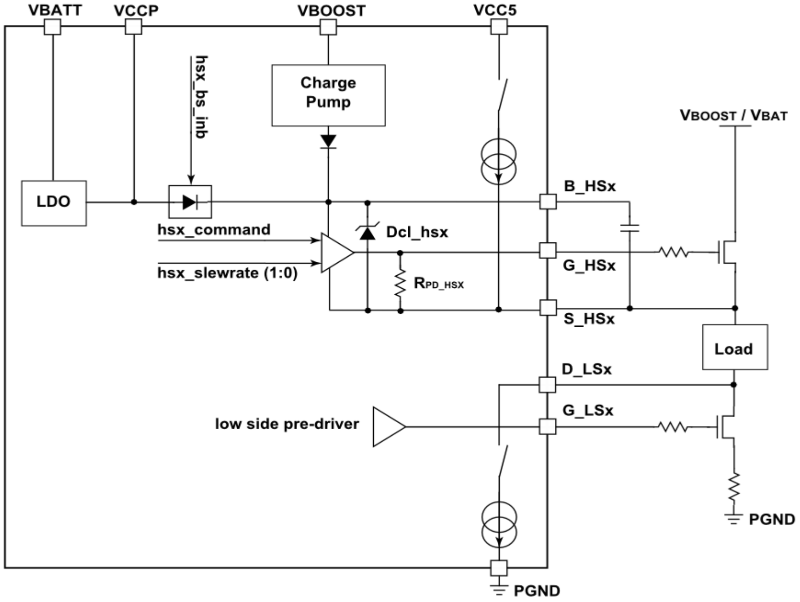


Figure 13. High-side Pre-driver Block Diagram

The high-side pre-driver is intended to drive the gate of an external logic level MOSFET in a high-side configuration.The logic command, hsx_command, to switch the external MOSFET, is provided by the microcores. This command is generated for taking into account the following signals:

- In the high-side pre-driver block, the signal DrvEn issued from the DRVEN pin is added to the control signal for the driver. As long as the DrvEn signal is low, the high-side pre-driver is switched off. The high-side pre-driver 5 includes a feature to override the switch off path via the DrvEn signal. As long as the hs5_ls36_en_ovr bit of the Driver_status register (0x1C5) is set to ‘1’, the pre-drivers are not influenced by DrvEn.
- VCCP undervoltage signals (uv_vccp) issued from the V_{CCP} monitoring: In case of an undervoltage, the external MOSFET is switched off.
- VCC5 undervoltage signals (uv_vcc5) issued from the V_{CC5} monitor: In case of an undervoltage, the external MOSFET is switched off.
- VBOOST undervoltage signals (uv_vboost) issued from the boost voltage monitor: In case of an undervoltage, the external MOSFET is switched off when this feature is enabled.
- Signal cksys_drven issued from the clock monitor. In case of a missing clock (PLL not locked), the external MOSFET is switched off. This function is disabled by default and can be enabled by setting the cksys_missing_disable_driver bit high in the Backup_clock_status_reg register r(0x1C7)
- Logic commands issued from logic channels (hsx_in).

The truth table describing the status of hsx_command signal is given in Table 28.

Table 28. High-side Pre-drivers Truth Table

DrvEn	uv_vccp	uv_vcc5	uv_vboost	cksys_drven	hsx_in	hsx_command	Driver status
0	–	–	–	–	–	0(40)	off
–	1	–	–	–	–	0(40)	off
–	–	1	–	–	–	0(40)	off
–	–	–	1	–	–	0(40)	off
–	–	–	–	0	–	0(40)	off
–	–	–	–	–	0	0(40)	off
1	0	0	0	1	1	1(41)	on

Note
40. When hsx_command is low, the G_HSx pin is driven low (pull-down to PGND voltage)
41. When hsx_command is high, the G_HSx pin is driven high (pull-up to V_{CCP} voltage)

Table 29. High-side Pre-drivers ON State Electrical Specifications

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
V_{S_HSX}	S_HSX pin operating voltage	-3.0	–	$V_{BOOSTMAX}$	V	(42)
	Transients t < 400 ns	-8.0	–	–	V	(42)
	Transients t < 800 ns	-6.0	–		V	(42)
V_{B_HSX}	B_HSX pin operating voltage	V_{S_HSX} + 4	–	V_{S_HSX} + 8	V	(42)
$V_{BS\_HSX\_CL}$	B_HSX to S_HSX voltage clamp, 15 µA < I_{CL} < 1.0 mA	6.5	7.3	8.0	V	
$V_{BS\_HS\_CL\_THREHOLD}$	B_HSX to S_HSX voltage threshold for hsx_cl_act internal signal activation	6.5	7.2	7.9	V	
V_{G_HSX}	G_HSX operating voltage	V_{S_HSX}	–	V_{B_HSX}	V	(42)
$I_{S\_HSX\_SINK}$	S_HSX leakage current, biasing switched off • V_{S_HS} = $V_{BOOSTMAX}$ • V_{S_HS} = 13.5 V • VS_HS = 7.0 V • VS_HS = 4.0 V	 – – – –	 – – – –	 1000 250 120 100	µA	
$I_{S\_HSX\_SINK\_ON}$	HSX leakage current when pre-driver on (biasing switched off) • V_{S_HS} = 7.0 V	 –	 –	 220	µA	
$I_{S\_HSX\_SINK\_DELTA}$	S_HSX leakage current delta between pre-drivers off and on • V_{S_HS} = 7.0 V	 60	 –	 140	µA	
I_{HSX_SUPL}	High-side driver supply current during 100% DC • During constant off • During constant on, including maximum supply current for the R_{PD_HSX} pull-down resistor	 – –	 – –	 30 25	µA	
$f_{G\_HSX\_PWM}$	PWM frequency • External $V_{CCP} \geq 6.5$ V max. • 9.0 V ≤ V_{BATT} • 5 V ≤ V_{BATT} ≤ 9.0 V	 0.0 0.0 0.0	 – – –	 100 100 50	KHz	(42)(43) (44)
DC_{G_HSX}	Duty cycle	0.0	–	100	%	(42)

Table 29. High-side Pre-drivers ON State Electrical Specifications (continued)

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$t_{ON\_HSX\_MIN}$	High-side driver minimum PWM on time	–	–	1.0	µs	(42)
$I_{G\_HSX\_PWM}$	G_HSx current (average during PWM operation) $Q_G = Q_{G\_HSX}$; f_{PWM} = 100 kHz	–	4.0	5.0	mA	(42)
$I_{G\_HSX\_SRC}$	Peak source gate drive current at fastest slew rate setting and VCCP = V_{GS} = 7.0 V, considering 10% and 90% of the output voltage.	–	230	–	mA	(42)
$I_{G\_HSX\_SRC}$	Peak sink gate drive current at fastest slew rate setting and VCCP = V_{GS} = 7.0 V, considering 10% and 90% of the output voltage.	–	440	–	mA	(42)
$t_{R\_G\_HSX}$	Turn on rise time, 10%–90% of out voltage; V_{CCP} = 7.0 V; at open pin	4.5	–	25	ns	(42)
$t_{F\_G\_HSX}$	Turn off fall time, 90%–10% of out voltage; V_{CCP} = 7.0 V; at open pin	5.0	–	25	ns	(42)
SR_{S_HSX}	Max permissible slew rate at the S_HSX pin. With higher slew rates, there may be a malfunction of the level shifter for the slew rate control.	-125	–	600	V/µs	(42)
$t_{DON\_G\_HSX}$	Turn on propagation delay at 300 V/µs slew rate; 10% of out voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	40	–	100	ns	(42)
$t_{DOFF\_G\_HSX}$	Turn off propagation delay at 300 V/µs slew rate; 10% of out voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	40	–	100	ns	(42)
$t_{DON\_G\_HSX}$	Turn on propagation delay at 50 V/µs slew rate; 10% of out voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	65	–	125	ns	(42)
$t_{DOFF\_G\_HSX}$	Turn off propagation delay at 50 V/µs slew rate; 10% of out voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	50	–	100	ns	(42)
$t_{DON\_G\_HSX}$	Turn on propagation delay at 25 V/µs slew rate;10% of out voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	100	–	200	ns	(42)
$t_{DOFF\_G\_HSX}$	Turn off propagation delay at 25 V/µs slew rate; 10% of out voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	70	–	150	ns	(42)
$t_{DON\_G\_HSX}$	Turn on propagation delay at 12.5 V/µs slew rate; 10% of out voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	160	–	310	ns	(42)
$t_{DOFF\_G\_HSX}$	Turn off propagation delay at 12.5 V/µs slew rate; 10% of out voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	90	–	170	ns	(42)

Note

42. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.
43. The minimum admitted series resistor is 0 Ω.
44. The external low-side MOSFET gate charge must not exceed 50 nC.A gate charge of maximum 75 nC is admitted if the $f_{PWM} \leq$ 67 kHz.

Table 30. High-side Pre-drivers OFF State Electrical Specifications

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
R_{PD_HSX}	G_HSX to S_HSX pull-down resistor • T_J = -40 to 150 °C	500	–	2000	kΩ	

6.3.1 High-side Driver Slew Rate Control

The driver strength can be selected individually for each of the drivers among a set of values by the SPI registers. There are four selectable driver strengths. The strength for the rising and falling edge can be chosen individually for each driver. Changing the rising edge affects the falling edge such as to retain the same absolute slew rate.

Table 31. High-side Pre-drivers Slew Rate Settings

hsx_slewrate_n(1:0)	Slew Rate (V/µs)	RDSON_PMOS (switching ON) (Ohm)	RDSON_NMOS (switching OFF) (Ohm)
00	300	14.6	5.9
01	50	84	35
10	25	170	69
11	12.5	337	138

The slew rates are selected by the SPI, writing the Hs_slew rate register (0x18E) while in normal mode (at reset we are in normal mode), but with the possibility of rapidly changing to the highest slew rate with a microcode instruction (*stslew*).

Table 32. High-side Pre-drivers Slew Rates Characteristics

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$R_{DS\_HSX\_P}$ (00)	G_HSx pMOS R_{DS_ON} (00), 300 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 2.5 V • at external V_{GS} = 1.0 to 4.0 V;	8.6 7.5	14.6 –	25.8 31.4	Ω	
$R_{DS\_HSX\_N}$ (00)	G_HSx nMOS R_{DS_ON} (00), 300 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 2.5 V • at external V_{GS} = 1.0 to 4.0 V	3.2 2.5	5.9 –	11.4 16.5	Ω	
$R_{DS\_HSX\_P}$ (01)	G_HSx pMOS R_{DS_ON} (01), 50 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 1.0 to 4.0 V	61	85	115	Ω	
$R_{DS\_HSX\_N}$ (01)	G_HSx nMOS R_{DS_ON} (01), 50 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 1.0 to 4.0 V	23	35	50	Ω	
$R_{DS\_HSX\_P}$ (10)	G_HSx pMOS R_{DS_ON} (10), 25 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 1.0 to 4.0 V	122	169	230	Ω	
$R_{DS\_HSX\_N}$ (10)	G_HSx nMOS R_{DS_ON} (10), 25 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 1.0 to 4.0 V	47	69	100	Ω	
$R_{DS\_HSX\_P}$ (11)	G_HSx pMOS R_{DS_ON} (11), 12.5 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 1.0 to 4.0 V	245	337	460	Ω	
$R_{DS\_HSX\_N}$ (11)	G_HSx nMOS R_{DS_ON} (11), 12.5 V/µs, V_{CCP} = 7.0 V at external V_{GS} = 1.0 to 4.0 V	94	138	199	Ω	
t_{SLR_HS}	Slew rate switching time • 1 ck cycle at 6.0 MHz, switching from slow to fast • 4 ck cycles at 6.0 MHz, witching from fast to slow	– –	– –	166 666	ns	(45)

Note

45. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.3.2 Bootstrap Diodes

"Bootstrapping" is a common way to create sufficient gate drive voltage for a high-side N-channel MOSFET. The charge needed to enhance the MOSFET's gate is stored in a bootstrap capacitor referenced to the MOSFET's source terminal. This method requires the bootstrap capacitor be charged through a low-side switch, current source or freewheeling diode that periodically pulls the negative terminal of the bootstrap capacitor to a voltage near to ground. Bootstrapping has the advantage of being simple and low cost, but it creates some operating limits, that is, the requirement to refresh the charge in the bootstrap capacitor limits the duty cycle and on-time.

The 33816 uses individual bootstrap circuits for each of its five high-side drivers. The bootstrap capacitor C_{B_HSX} for each high-side driver is directly charged from V_{CCP} through a bootstrap diode as soon as the voltage on high-side MOSFET source pin drops to a voltage close to 0.0 V and therefore the V_{B_HSX} voltage drops below the V_{CCP} voltage.

The bootstrap control circuitry:

- Turns on the bootstrap diode to load the bootstrap capacitor when the high-side driver's source terminal is significantly below the V_{CCP} voltage
- Clamps the high-side gate voltage when the bootstrap capacitor is reaches the desired voltage
- Prevents bootstrap capacitor discharge when the B_HSX pin voltage is higher than V_{CCP}.

Bootstrap operation can create charging currents sufficiently large that injector diagnostics can be affected. To avoid such disturbances, the digital core issues one signal for each high-side pre-driver (called hsx_bs_inb) that prevents the bootstrap switches from switching on at the end of the injection or during initialization, as long as no low-side pre-driver is switched on. It thereby prevents degradation of end of the injection monitoring performed when using the V_{SRC} comparator of the pre-driver.

Current flowing through each bootstrap diode is actively limited to avoid overloading them.

Table 33. Bootstrap Diode Electrical Specifications

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
I_{DB_HSX}	Bootstrap path current capability at 6.5 V < VCCP < 7.5 V • V_{CCP} - V_{B_HSX} = 0.25 V • V_{CCP} - V_{B_HSX} = 0.5 V • V_{CCP} - V_{B_HSX} = 0.75 V • V_{CCP} - V_{B_HSX} = 1.0 V • V_{CCP} - V_{B_HSX} = 1.5 V • V_{CCP} - V_{B_HSX} = 2.0 V • V_{CCP} - V_{B_HSX} > 2.0 V	 -38 -67 -83 -78 -79 -84 -95	 – – – – – – –	 -9.1 -18 -29 -39 -42 -43 -47	mA	
R_{DB_HSX}	Bootstrap path resistance • V_{CCP} - V_{B_HSX} < 1.0 V	–	–	30	Ω	
$V_{B\_HSX\_VCCP\_\_TH\_R}$	Bootstrap path V_{B_HSX} to V_{CCP} voltage threshold when bootstrap voltage rising	-150	–	-30	mV	
$V_{B\_HSX\_VCCP\_\_TH\_F}$	Bootstrap path V_{B_HSX} – V_{CCP} voltage threshold when bootstrap voltage falling	-165	–	-40	mV	
$V_{B\_HSX\_VCCP\_\_TH\_HYST}$	Bootstrap path V_{B_HSX} – V_{CCP} threshold hysteresis	0.0	–	30	mV	
$t_{DB\_HSX\_ON}$	Bootstrap switch turn on delay during PWM • Delay from VB_HSx < VCCP to bootstrap switch ON including comparator delay and CP charge time.	–	–	450	ns	(46)
$t_{MAXB\_HSX\_ON}$	Bootstrap switch on time after switching • After this time the bootstrap switch NMOS transistor is switched off due to gate discharge currents.	100	–	ms	ms	(46)

Table 33. Bootstrap Diode Electrical Specifications

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$I_{DB\_HSX\_LOW}$	Bootstrap low current limit	-570	-375	-280	µA	

Note

46. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.3.2.1 Bootstrap Start-up Default Sequence

The typical way to startup the bootstrap capacitor for each high-side pre-driver requires the topology shown in Figure 14.

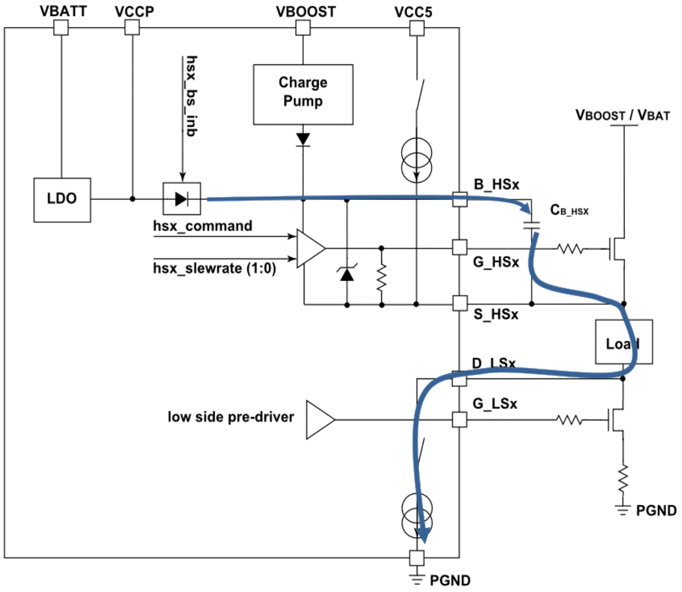


Figure 14. Booststrap Load Current Path During Start-up Default Sequence

The DBG pin must be set high or unconnected, as a weak high-side pull-up is internally implemented when the device was in reset.

After reset release and after the VCCP voltage is above the $V_{UVVCCP+}$ threshold, all low-side biasings are activated and the charging of the high-side pre-driver's bootstrap capacitors via the D_LSx pull-down sources starts automatically, as long as there is a current path from S_HSx of the high-side pre-driver to at least one D_LSx pin or to GND.

During this phase for each of the five high-side driver pre-drivers:

- The bootstrap diode current is limited
- The V_{SRC} threshold is forced, so it cannot be used during this phase
- All low-side biasings are activated
- All high-side biasings are disabled.

This initialization phase is interrupted if one of the following conditions is reached individually for each high-side pre-driver:

- B_HSx voltage is close to V_{CCP} voltage (typically 7.0 V) and the S_HSx voltage is below 0.5 V, in a range of 36 ms after the V_{CCP} undervoltage threshold $V_{UVVCCP+}$ was exceeded, then 1.0 V
- The bootstrap voltage clamp is active and the S_HSx voltage is below 0.5 V, in a range of 36 ms after the V_{CCP} undervoltage threshold $V_{UVVCCP+}$ was exceeded, then 1.0 V
- The low-side pre-driver affected to the high-side pre-driver is switched on by the microcores or by SPI

- The low-side pre-driver to the high-side pre-driver association is disabled
- The high-side pre-driver switched on

The association described by the following is configured through the Hs12_ls_act register (0x1A6), the Hs34_ls_act register (0x1A7), and the Hs5_ls_act register (0x1A8).

If two high-side pre-drivers are affected by a unique low-side pre-driver, the suitable blanking times must be applied to avoid unexpected interruption of the initialization phase from being interrupted before completion. During this blanking time none of the two high-side biasing must be turned on by the SPI or microcore.

Table 34. Bootstrap Charge Time Up To 7.0 V During Start-up Default Sequence

Bootstrap Capacitor Typical Value	Typical Charge Time (ms)
100 nF	2.3
300 nF	7.7
1.0 µF	23.3
2.2 µF	51.3

It is recommended to turn off the initialization phase for each unused high-side pre-drivers, by removing any low-side pre-driver association in the Hs12_ls_act register (0x1A6), the Hs34_ls_act register (0x1A7), and the Hs5_ls_act register (0x1A8).

6.3.2.2 Bootstrap Start-up Sequence Using the Charge Pump

If there is no current path from S_HSx pin to D_LSx or GND, the internal charge pump can be used to charge the bootstrap capacitors during initialization, considering a consumption of 20 µA (IHSX_SUPL) per high-side pre-driver. In this case, no additional current must be drawn from the B_HSx pin.

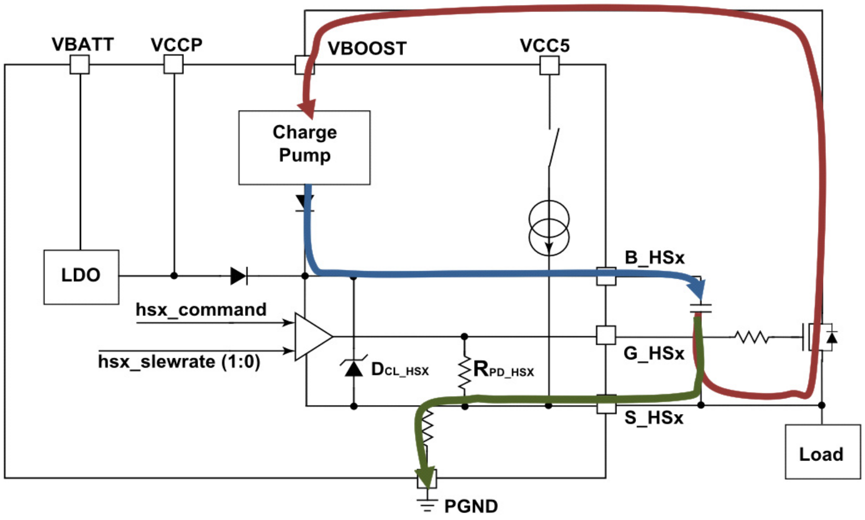


Figure 15. Booststrap Load Current Path During Start-up Sequence Using Charge Pump

Turning on either the low-side MOSFET or the D_LSx pull-down current sources to charge the bootstrap capacitors during the initialization phase is not required, if the current loop exists via the body diode of the external high-side MOSFET. In addition, there is some leakage current ($I_{S\_HSX\_SINK}$) path from S_HSx to PGND. The charge pump starts charging the bootstrap capacitors as soon as the device is supplied with VCC5, and the VBOOST pin voltage is greater than 4.7 V, and POResetB is deactivated.

Table 35. Bootstrap Charge Time Up To 7V Using Charge Pump

Bootstrap Capacitor Typical Value	Typical Charge Time (ms)
100 nF	35
300 nF	116
1.0 µF	350
2.2 µF	770

Using a charge pump to augment the bootstrap function does two things: it eliminates most of the duty cycle and on-time limitations, and it can pre-charge the bootstrap capacitor at power up.

6.3.3 Charge Pump

The 33816 provides one charge pump with independent outputs for each of the five high-side drivers. The independent outputs allow complete flexibility of the topology used, that is, the D_HSX pins of the high-side channels can be connected to different voltage levels (for example, V_{BAT} or V_{BOOST}).

In most operating topologies and conditions, the bootstrap diode is the primary source of charge for the bootstrap capacitor, and the charge pump sustains the voltage at each bootstrap capacitor when it is not being charged by low-side switching.

This charge pump allows 100% duty cycle operation of the high-side MOSFETs while the bootstrap circuitry is not operating (VS_HSx voltage never goes significantly below the V_{CCP} voltage). In that condition, the charge pump provides current that maintains each bootstrap capacitor charged via independent current sources, to guarantee a minimum V_{GS} voltage.

The charge pump, supplied by VBOOST, creates gate drive voltages about 8.0 V greater than the voltage at VBOOST. However, their current capacity is sufficient only for low frequency switching. in addition, VCC5 supplies the charge pump circuitry.

Each current source charges its bootstrap capacitor as long as the voltage at its B_HSX pin is less than the charge pump output voltage. The current stops when B_HSX exceeds the charge pump output voltage. The individual current sources supply the high-side pre-driver and its MOSFET's gate, charges the bootstrap capacitor, or flows through the bootstrap clamp (D_{CL_HSX}), if the bootstrap capacitor is already fully charged.

All high-side pre-drivers are disabled when the voltage at the VBOOST pin is less than its undervoltage lockout threshold, which is around 4.7 V.

The charge pump is not running as long as the POResetB reset signal is active.

Table 36. Charge Pump Electrical Specifications

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
V_{CP}	Charge pump output voltage	–	–	V_{BOOST} + 8	V	
I_{CP}	Charge pump output current capability	375	–	–	µA	
f_{CP}	Charge pump clock frequency	–	28	–	MHz	
t_{CP_init}	Time for Charge Pump Initialization	–	10	–	µs	(47)
$I_{CP\_SR\_MAX}$	Charge pump individual output current source current capability	50	–	62	µA	
V_{CP_SRC}	Charge pump current source output voltage, output voltage at B_HSX; 0.0 V ≤ V_S_HSX ≤ V_{BOOST} • V_{BOOST} > 4.7 V; V_{S_HSx} = 4.7 V • V_{BOOST} > 6.0 V; V_{S_HSx} = 6.0 V • V_{BOOST} > 72 V; V_{S_HSx} = 72 V	 V_{S_HSX} + 4 V_{S_HSX} + 6 V_{S_HSX} + 6	 – – –	 – – –	V	

Note

47. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.3.4 Safe State of High-side Pre-driver

In order to guarantee a safe condition while the device is not operating, the G_HSX output is immediately forced to the low level, switching the external MOSFET off when reset (RSTB) is asserted, or DrvEn is low. This behavior is effective as long as the bootstrap capacitor voltage is greater than a typical voltage of 1.1 V.

When the bootstrap capacitor voltage is lower than or equal to 1.1 V, the pre-driver output state is undefined, but the pre-driver is not in a high state. In addition, an integrated pull-down resistor R_{PD_HSX} between G_HSX and S_HSX keeps the external MOSFET in an OFF state.

6.3.5 High-side Pre-drivers in Low-side Configuration

All high-side pre-drivers can be used as low-side pre-drivers. In this configuration, an external booststrap capacitor is still required. However, the VDS monitoring for this low-side MOSFET is not functional.

6.4 Low-side Pre-drivers (LS1 - LS6)

6.4.1 General Description

The 33816 provide six independent low-side pre-drivers designed to drive the gates of external low-side configuration N-channel logic level MOSFETs. These pre-drivers are dedicated the load driving like injectors or solenoid and integrate diagnosis features.

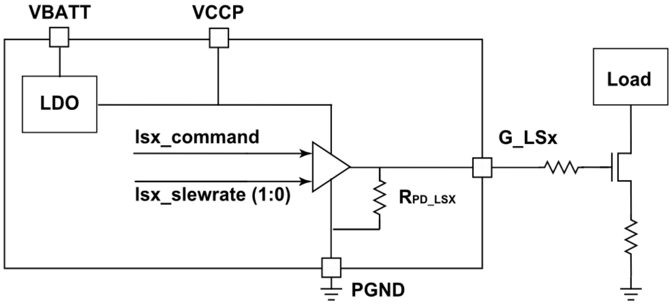


Figure 16. Low-side Pre-driver Block Diagram

Internal to the device, a gate to source pull-down resistor R_{PD_LSX} holds the external MOSFETs in the off state, while the device is in a power on reset state (RSTB low).

The low-side pre-drivers are supplied by V_{CCP} voltage.

The low-side pre-driver is intended to drive the gate of an external logic level MOSFET in low-side configuration. The logic command lsx_command, to switch the external MOSFET, is provided by the digital block. This command is generated, taking into account the following signals:

- The signal DrvEn is issued from the DRVEN pin. As long as the DrvEn signal is negated, the low-side pre-driver is switched off. The low-side pre-driver 3 and 6 includes a feature to override the switch off path via the DrvEn signal. As long as the hs5_ls36_en_ovr bit of the Driver_config register (0x1C5) is set to ‘1’, the pre-drivers are not influenced by DrvEn.
- The V_{CCP} undervoltage signals (uv_vccp) issued from the V_{CCP_UV} monitoring. In case of an undervoltage, the external MOSFET is switched off
- The V_{CC5} undervoltage signals (uv_vcc5) issued from the V_{CC5_UV} monitoring. In case of an undervoltage, the external MOSFET is switched off
- The cksys_drven signal issued from the clock monitoring. In the event of a missing clock, the external MOSFET is switched off. This condition can be optionally disabled.
- The logic command coming from channel logic (lsx_in)

The truth table describing the status of lsx_command signal is given in Table 37.

Table 37. Low-side Pre-driver Truth Table

DrvEn	uv_vccp	uv_vcc5	cksys_drven	lsx_in	lsx_command	Driver status
0	–	–	–	–	0(48)	off
–	1	–	–	–	0(48)	off
–	–	1	–	–	0(48)	off
–	–	–	0	–	0(48)	off
–	–	–	–	0	0(48)	off
0	0	0	0	1	1(49)	on

Note
48. When lsx_command is low, the G_LSx pin is driven low (pull-down to PGND voltage)
49. When lsx_command is high, the G_LSx pin is driven high (pull-up to V_{CCP} voltage)

Table 38. Low-side Pre-drivers ON State Electrical Specifications

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
V_{G_LSX}	G_LSX operating voltage	0.0	–	V_{CCP}	V	(50)
$I_{S\_LSX\_SINK}$	D_LSX leakage current (biasing switched off) • V_{D_LSx} = 13.5 V • V_{D_LSx} = 40 V	 10 10	 – –	 110 320	µA	
$f_{G\_LSX\_PWM}$	PWM frequency • Nominal • t < 50 µs	 0.0 0.0	 – –	 100 200	kHz	(50)(51) (52)
DC_{G_LSX}	Duty cycle	0.0	–	100	%	(50)
$I_{G\_LSX\_PWM}$	G_LSX current (average during PWM operation) • $Q_G = Q_{G\_LSX}$; f_{PWM} = 100 kHz	 –	 3.0	 5.0	mA	(50)
$I_{G\_LSX\_SRC}$	Peak source gate drive current at fastest slew rate setting and $V_{CCP} = V_{GS}$ = 7.0 V	–	230	–	mA	(50)
$I_{G\_LSX\_SRC}$	Peak sink gate drive current at fastest slew rate setting and $V_{CCP} = V_{GS}$ = 7.0 V	–	440	–	mA	(50)
$t_{R\_G\_LSX}$	Turn on rise time, 10%-90% of output voltage; V_{CCP} = 7.0 V; at Open pin	5.0	–	25	ns	(50)
$t_{F\_G\_LSX}$	Turn off fall time, 90%-10% of output voltage; V_{CCP} = 7.0 V; at Open pin	5.0	–	25	ns	(50)
$t_{DON\_G\_LSX}$	Turn on propagation delay at 300 V/µs slew rate; 10% of output voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	10	–	70	ns	(50)
$t_{DOFF\_G\_LSX}$	Turn off propagation delay at 300 V/µs slew rate; 10% of output voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	10	–	70	ns	(50)
$t_{DON\_G\_LSX}$	Turn on propagation delay at 50 V/µs slew rate; 10% of output voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	10	–	80	ns	(50)
$t_{DOFF\_G\_LSX}$	Turn off propagation delay at 50 V/µs slew rate; 10% of output voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	10	–	80	ns	(50)
$t_{DON\_G\_LSX}$	Turn on propagation delay at 25 V/µs slew rate;10% of output voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	15	–	120	ns	(50)
$t_{DOFF\_G\_LSX}$	Turn off propagation delay at 25 V/µs slew rate; 10% of output voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	15	–	120	ns	(50)

Table 38. Low-side Pre-drivers ON State Electrical Specifications (continued)

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$t_{DON\_G\_LSX}$	Turn on propagation delay at 12.5 V/µs slew rate; 10% of output voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	15	–	150	ns	(50)
$t_{DOFF\_G\_LSX}$	Turn off propagation delay at 12.5 V/µs slew rate; 10% of output voltage change; C_{LOAD} = 4.7 nF; R_G = 40.2 Ω; V_{CCP} = 7.0 V	15	–	150	ns	(50)

Note

50. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.
51. The minimum admitted series resistor is 0 Ω.
52. The external low-side MOSFET gate charge must not exceed 50 nC. A gate charge of maximum 75 nC is admitted if the $f_{PWM} \leq$ 67 kHz. A gate charge of maximum 100 nC is admitted if the $f_{PWM} \leq$ 50 kHz.

Table 39. Low-side Pre-drivers OFF State Electrical Specifications

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
R_{PD_LSX}	G_LSX to PGND pull-down resistor	25	50	90	kΩ	

6.4.2 Low-side Pre-driver Slew Rate Control

Each driver strength can be selected individually by the SPI registers within a set of values. There are four selectable driver strengths. The strength for the rising and falling edge can be individually chosen for each driver. A change in the rising edge affects the falling edge such as to keep the same absolute slew rate.

Table 40. Low-side Pre-drivers Slew Rate Settings

lsx_slewrate_n(1:0)	Slew Rate (V/µs)	RDSON_PMOS (switching ON) (Ohm)	RDSON_NMOS (switching OFF) (Ohm)
00	300	14.6	5.9
01	50	84	35
10	25	170	69
11	12.5	337	138

The slew rates are selected by the SPI, writing the Ls_slewrate register (0x18F) while in Normal mode (at reset we are in normal mode), but with the possibility of rapidly changing to the highest slew rate with the microcode instruction *stslew*.

Table 41. Low-side Pre-drivers Slew Rates Characteristics

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$R_{DS\_HSX\_P}$ (00)	G_HSx pMOS R_{DS_ON} (00), 300 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 2.5 V • at external V_{GS} = 1.0 to 4.0 V	8.8 7.5	14.6 –	25.3 31.3	Ω	
$R_{DS\_HSX\_N}$ (00)	G_HSx nMOS R_{DS_ON} (00), 300 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 2.5 V • at external V_{GS} = 1.0 to 4.0 V	3.4 2.5	5.9 –	11.1 16.5	Ω	
$R_{DS\_HSX\_P}$ (01)	G_HSx pMOS R_{DS_ON} (01), 50 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 1.0 to 4.0 V	61	84	115	Ω	
$R_{DS\_HSX\_N}$ (01)	G_HSx nMOS R_{DS_ON} (01), 50 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 1.0 to 4.0 V	23	35	50	Ω	
$R_{DS\_HSX\_P}$ (10)	G_HSx pMOS R_{DS_ON} (10), 25 V/µs, V_{CCP} = 7.0 V • at external = 1.0 to 4.0 V	122	170	230	Ω	
$R_{DS\_HSX\_N}$ (10)	G_HSx nMOS R_{DS_ON} (10), 25 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 1.0 to 4.0 V	47	69	100	Ω	
$R_{DS\_HSX\_P}$ (11)	G_HSx pMOS R_{DS_ON} (11), 12.5 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 1.0 to 4.0 V	245	337	460	Ω	
$R_{DS\_HSX\_N}$ (11)	G_HSx nMOS R_{DS_ON} (11), 12.5 V/µs, V_{CCP} = 7.0 V • at external V_{GS} = 1.0 to 4.0 V	94	138	199	Ω	
t_{SLR_HS}	Slew rate switching time • 1 ck cycle at 6.0 MHz	–	–	166	ns	(53)

Note

53. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size from typical devices under typical conditions, unless otherwise noted.

6.4.3 Safe State of Low-side Pre-driver

In order to guarantee a safe condition, the G_LSX output is immediately forced to a low level, switching off the external MOSFET when a reset is asserted and while the device is not operating. In this phase, the pre-driver is powered by the charge already stored in the VCCP buffer capacitor, and a low level output is guaranteed, as long as a typical voltage greater than 1.1 V is available.

When the V_{CCP} supply voltage is lower than 1.1 V, the pre-driver output is pulled to PGND by an internal high resistance R_{PD_LSX} pull-down resistor.

6.5 V_{DS} and V_{SRC} Monitor and Load Biasing

The 33816 provides a VDS monitoring function for diagnostic and protection for each of the five high-side pre-drivers, and for six of the low-side pre-divers. The LS7 pre-driver dedicated to the DC-DC converter does not integrate this diagnosis feature.

Moreover, a source voltage monitoring function V_{SRC} populates each of the five high-side pre-drivers.

The V_{DS} monitors measure:

- the voltage between the VBOOST or VBATT pin, and the source pin of the external MOSFET connected to the S_HSx device pin, for the HS2 and HS4 high-side pre-drivers
- the voltage between the VBATT pin and the source pin of the external MOSFET connected to the S_HSx device pin, for the HS1, HS3, and HS5 high-side pre-drivers
- the voltage between the drain pin of the external MOSFET connected to the D_LSx device pin and the PGND pin, for the low-side pre-drivers

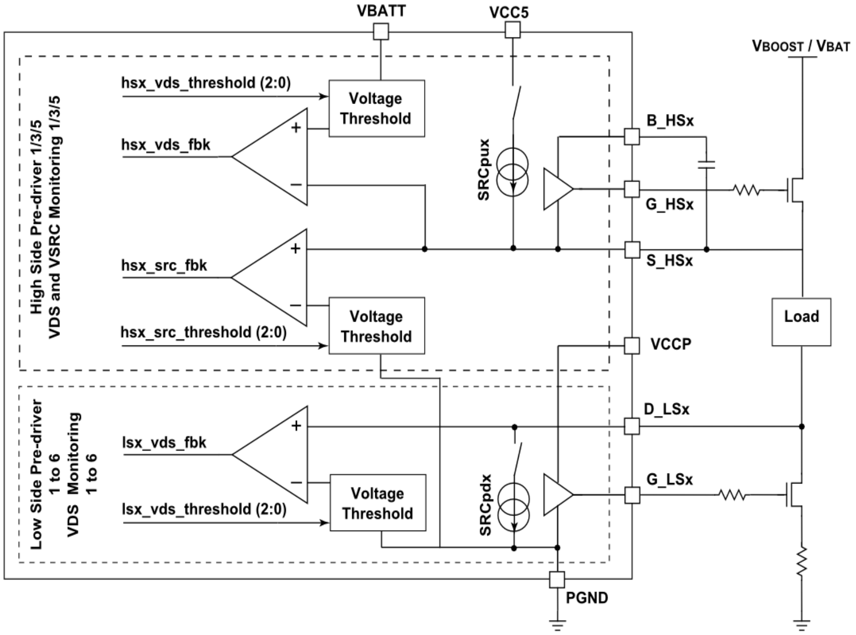


Figure 17. V_{DS} and V_{SRC} Monitors and Load Biasing

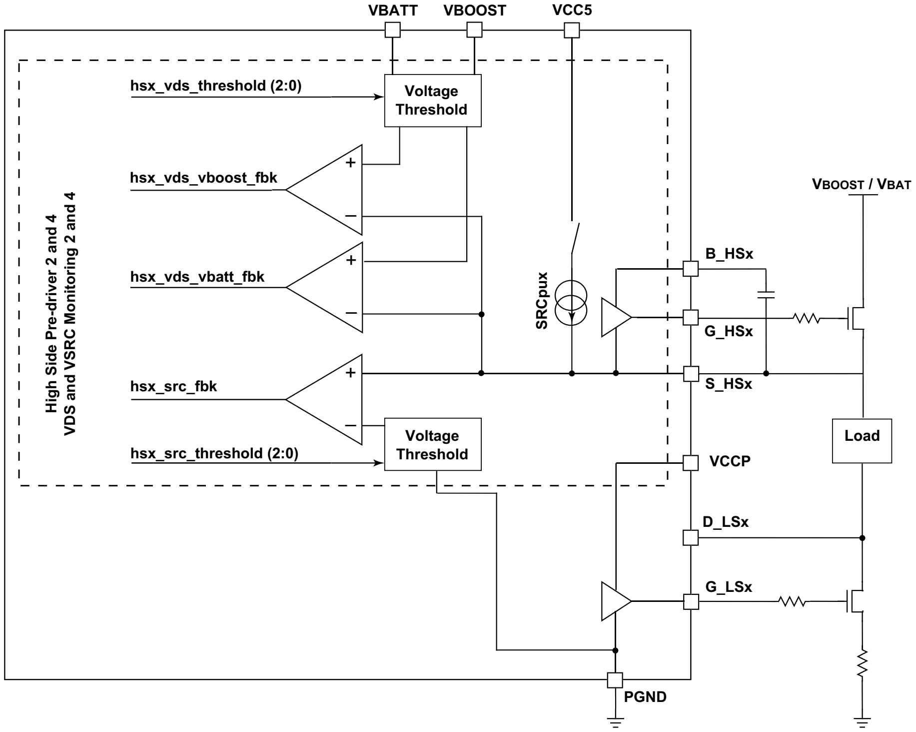


Figure 18. V_{DS} Monitors and Load Biasing for HS2 and HS4

The V_{SRC} monitors measure on the high-side pre-drivers, the voltage between the source pin of the external MOSFET connected to the S_HSx device pin and the PGND pin.

Their thresholds are individually selectable for each output by setting the suitable values in:

- the Vds_threshold_hs (0x18A) register for the high-side pre-drivers V_{DS} threshold
- the Vsrc_threshold_hs (0x18B) register for the high-side pre-drivers V_{SRC} threshold
- the Vds_threshlod_ls_1 (0x18C) and Vds_threshlod_ls_1 (0x18D) registers for the low-side pre-drivers V_{DS} threshold

These thresholds are selectable either by the SPI or by microcode (*chth* instruction).

The V_{DS} and V_{SRC} monitor functions are available, since the corresponding pre-drivers are supplied.

The high-side V_{DS} and V_{SRC} monitors can work in standalone mode without using the associated high-side pre-driver. In this case, the B_HSX and G_HSX outputs is not connected and the corresponding high-side driver cannot be used. The S_HSX input can be connected to any node, as long as the maximum is within the pin’s maximum rating range.

6.5.1 High-side V_{DS} And V_{SRC} Monitoring

The high-side V_{DS} and V_{SRC} monitors are functionally independent from the bootstrap voltage of the high-side pre-driver.

The two high-side V_{DS} monitors of pre-driver 2 and 4 are composed of three comparators with programmable thresholds, the first one senses the voltage between VBOOST and the S_HSX source pin (V_{DS} of the high-side MOSFET used as boost MOSFET). The second one senses the voltage between VBATT and the S_HSX source pin (V_{DS} of the high-side MOSFET, used as a battery MOSFET and voltage information for voltage based diagnosis, when the MOSFET is in boost configuration) The third one senses the voltage between the S_HSX source pin and PGND (voltage across the freewheeling element, either a diode or a MOSFET).

The instruction *slfbk* allows to select which of the two feedbacks hsx_vds_vboost_fbk or hsx_vds_vbatt_fbk, enables the microcores.

Two voltage references per high-side pre-driver provide a voltage threshold to the V_{DS} comparators. Their values are selectable among eight values, according to the hsx_vds_threshold(2:0) and the hsx_src_threshold(2:0) signal, provided by the digital cores.

The current values of the hsx_vds_threshold(2:0) and the hsx_src_threshold(2:0) are programmed through the SPI by accessing the Vds_regfile registers (0x18A and 0x18B).

Table 42. V_{DS} and V_{SRC} Monitoring Typical Threshold Selection for High-side Pre-drivers

hsx_vds/src_ threshold(2:0)	VDS (V)	VSRC (V)
000	0.00	0.0
001	0.5	0.5
010	1	1
011	1.5	1.5
100	2.0	2.0
101	2.45	2.5
110	2.95	3.0
111	3.45	3.5

In case a fast dv/dt is applied (i.e. the SR_{S_HSX} maximum value is exceeded) at the S_HSX pin (for instance, during boost MOSFET commutations), the comparator output may have an incorrect value while the disturbance is applied. The function recovers after the disturbance removal to the nominal behavior in less than a typical 300 ns.

During freewheeling operation, the S_HSX source pin, can go down to a typical -3.0 V, and can withstand transients of -6.0 V/-8.0 V for a duration shorter than 400 ns at a very high MOSFET switch.

Table 43. High-side VDS and VSRC Monitoring Electrical Specifications

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$V_{S\_HS\_VDS}$	High-side $V_{DS/SRC}$ monitoring functional range S_HSx • Transients t < 400 ns • Transients t < 800 ns	-3.0 -8.0 -6.0	– – –	72 – –	V	(54)
V_{VBATT_VDS}	High-side $V_{DS/SRC}$ monitoring functional range VBATT • $V_{DS\_HS\_TH}$ 3.5 V is at 3.0 V min.	5.5 5.0	– –	72 5.5	V	
V_{VBOOST_VDS}	High-side $V_{DS/SRC}$ monitoring functional range VBOOST • $V_{DS\_HS\_TH}$ 3.5 V is at 3.0 V min.	5.5 5.0	– –	72 5.5	V	
$V_{DS\_HS\_TH}$	High-side V_{DS} threshold (000) • V_{VBATT_VDS} = 5.0 V to 72 V • V_{VBOOST_VDS} = 5.0 V to 72 V	-0.1	0.0	0.1	V	
$V_{DS\_HS\_TH}$	High-side V_{DS} threshold (001) • V_{VBATT_VDS} = 5.0 V to 72 V • V_{VBOOST_VDS} = 5.0 V to 72 V	0.4	0.5	0.6	V	
$V_{DS\_HS\_TH}$	High-side V_{DS} threshold (010) • V_{VBATT_VDS} = 5.0 V to 72 V • V_{VBOOST_VDS} = 5.0 V to 72 V	0.9	1.0	1.1	V	
$V_{DS\_HS\_TH}$	High-side V_{DS} threshold (011) • V_{VBATT_VDS} = 5.0 V to 72 V • V_{VBOOST_VDS} = 5.0 V to 72 V	1.35	1.5	1.65	V	
$V_{DS\_HS\_TH}$	High-side V_{DS} threshold (100) • V_{VBATT_VDS} = 5.0 V to 72 V • V_{VBOOST_VDS} = 5.0 V to 72 V	1.8	2.0	2.2	V	
$V_{DS\_HS\_TH}$	High-side V_{DS} threshold (101) • V_{VBATT_VDS} = 5.0 V to 72 V • V_{VBOOST_VDS} = 5.0 V to 72 V	2.29	2.45	2.61	V	
$V_{DS\_HS\_TH}$	High-side V_{DS} threshold (110) • V_{VBATT_VDS} = 5.0 V to 72 V • V_{VBOOST_VDS} = 5.0 V to 72 V	2.76	2.95	3.14	V	
$V_{DS\_HS\_TH}$	High-side V_{DS} threshold (111) • V_{VBATT_VDS} = 5.5 V to 72 V • V_{VBOOST_VDS} = 5.5 V to 72 V • V_{VBATT_VDS} = 5.0 V to 5.5 V • V_{VBOOST_VDS} = 5.0 V to 5.5 V	3.23 3.23 3.0 3.0	3.5 3.5 3.45 3.45	3.67 3.67 3.67 3.67	V	
t_{TH_HSVDS}	High-side $V_{DS/SRC}$ threshold settling time • From HS_VDS/SRC_threshold(2:0) change to threshold stable	–	0.4	1.0	µs	(54)
t_{D_HSVDS}	High-side $V_{DS/SRC}$ comparator switching time, Propagation delay + rise/fall time • At 100 mV overdrive • At 200 mV overdrive • At 300 mV overdrive	– – –	1.0 0.8 0.6	1.5 1.1 1.0	µs	(54)(55)
t_{R_HSVDS}	High-side $V_{DS/SRC}$ comparator recovery time after dV_D/dt • Recovery time after dV_D/dt is removed	–	–	300	ns	(54)
SR_{HSVDS}	High-side $V_{DS/SRC}$ comp. input voltage slew rate with good output at VDS_HS=0.75 V with VS_HSx from 13.5 V to 72 V and from 72 V to 13.5 V	–	–	100	V/µs	(54)

Table 43. High-side VDS and VSRC Monitoring Electrical Specifications (continued)

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$V_{SRC\_HS\_TH}$	High-side V_{SRC} threshold (000)	-0.1	0.0	0.1	V	
$V_{SRC\_HS\_TH}$	High-side V_{SRC} threshold (001)	0.4	0.5	0.6	V	
$V_{SRC\_HS\_TH}$	High-side V_{SRC} threshold (010)	0.9	1.0	1.1	V	
$V_{SRC\_HS\_TH}$	High-side V_{SRC} threshold (011)	1.35	1.5	1.65	V	
$V_{SRC\_HS\_TH}$	High-side V_{SRC} threshold (100)	1.8	2.0	2.2	V	
$V_{SRC\_HS\_TH}$	High-side V_{SRC} threshold (101)	2.38	2.55	2.72	V	
$V_{SRC\_HS\_TH}$	High-side V_{SRC} threshold (110)	2.85	3.0	3.15	V	
$V_{SRC\_HS\_TH}$	High-side V_{SRC} threshold (111)	3.33	3.5	3.68	V	
$V_{SRC\_HS\_Th}$	High-side V_{SRC} threshold (000) including crosstalk	-0.2	0.0	0.2	V	(54)(56)
$V_{SRC\_HS\_Th}$	High-side V_{SRC} threshold (001) including crosstalk	0.3	0.5	0.7	V	(54)(56)
$V_{SRC\_HS\_Th}$	High-side V_{SRC} threshold (010) including crosstalk	0.8	1.0	1.2	V	(54)(56)
$V_{SRC\_HS\_Th}$	High-side V_{SRC} threshold (011) including crosstalk	1.25	1.5	1.75	V	(54)(56)
$V_{SRC\_HS\_Th}$	High-side V_{SRC} threshold (100) including crosstalk	1.7	2.0	2.3	V	(54)(56)
$V_{SRC\_HS\_Th}$	High-side V_{SRC} threshold (101) including crosstalk	2.28	2.55	2.82	V	(54)(56)
$V_{SRC\_HS\_Th}$	High-side V_{SRC} threshold (110) including crosstalk	2.75	3.0	3.25	V	(54)(56)
$V_{SRC\_HS\_Th}$	High-side V_{SRC} threshold (111) including crosstalk	3.23	3.5	3.78	V	(54)(56)

Note

54. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.
55. The high value is a worst case consideration on an overdrive of 100 mV.
56. To limit the amplitude of the crosstalk to the specified value, the sequence of monitor threshold switching must not exceed a certain number of switches in any given time window. Any sequence of n commands which increase a threshold of a low-side VDS or high-side VSRC monitor must be spread over a time window t_{N_I} which meets the condition: $t_{N\_I} \geq (N - 7) * 1.67$ µs. Any sequence of m commands which decrease a threshold must be spread over a time window t_{M_D} which meets the condition: $t_{M\_D} \geq (M - 7) * 1.67$ µs.

6.5.2 Low-side V_{DS} Monitoring

A comparator with a programmable threshold is provided for V_{DS} monitoring of the external low-side MOSFET, sensing the voltage between the D_LSx drain pin and PGND (VDS of the low-side MOSFET). If a sense resistor is connected between the low-side MOSFET and ground, the voltage drop on the resistor is included in the measurement.

One voltage reference per low-side pre-driver provides a voltage threshold to the V_{DS} comparator. Its value is selectable among eight values, according to the lsx_vds_threshold(2:0) signal, provided by the digital cores.

The current values of lsx_vds_threshold(2:0) are programmed through the SPI by accessing the Vds_regfile registers (0x18C and 0x18D).

Table 44. VDS Monitoring Typical Threshold Selection for Low-side Pre-drivers

lsx_vds threshold(2:0)	VDS (V)
000	0.00
001	0.5
010	1.0
011	1.5
100	2.0
101	2.5
110	3.0
111	3.5

If a fast dv/dt is applied at the D_LSX pin (for instance, after a fast decay), the comparator output may have an incorrect value. When a disturbance is applied, the function recovers from the disturbance removal to a nominal behavior in less than a typical 300 ns.

The operating voltage range of low-side MOSFET D_LSX drain pin is up to 75 V. During freewheeling operation on one load, the D_LSX source pin of a different load connected to the same bank, can go down to a typical -3.0 V.

The low-side V_{DS} monitors can even be used standalone, without using the associated low-side pre-driver. In this case, the G_LSX output is not connected. The D_LSX input can be connected to any node within the pin maximum ratings voltage range.

Table 45. Low-side VDS Monitoring Electrical Specifications

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$V_{D\_LSX\_VDS}$	Low-side V_{DS} monitoring functional range D_LSx • Transients t < 400 ns	-3.0 -8.0	– –	75 –	V	(57)
$V_{DS\_LS\_TH}$	Low-side V_{DS} threshold (000)	-0.1	0.0	0.1	V	
$V_{DS\_LS\_TH}$	Low-side V_{DS} threshold (001)	0.4	0.5	0.6	V	
$V_{DS\_LS\_TH}$	Low-side V_{DS} threshold (010)	0.9	1.0	1.1	V	
$V_{DS\_LS\_TH}$	Low-side V_{DS} threshold (011)	1.35	1.5	1.65	V	
$V_{DS\_LS\_TH}$	Low-side V_{DS} threshold (100)	1.8	2.0	2.2	V	
$V_{DS\_LS\_TH}$	Low-side V_{DS} threshold (101)	2.38	2.5	2.63	V	
$V_{DS\_LS\_TH}$	Low-side V_{DS} threshold (110)	2.85	3.0	3.15	V	
$V_{DS\_LS\_TH}$	Low-side V_{DS} threshold (111)	3.33	3.5	3.68	V	
$V_{DS\_LS\_Th}$	Low-side V_{DS} threshold (000) incl. crosstalk	-0.2	0.0	0.2	V	(57)(58)
$V_{DS\_LS\_Th}$	Low-side V_{DS} threshold (001) incl. crosstalk	0.3	0.5	0.7	V	(57)(58)
$V_{DS\_LS\_Th}$	Low-side V_{DS} threshold (010) incl. crosstalk	0.8	1.0	1.2	V	(57)(58)
$V_{DS\_LS\_Th}$	Low-side V_{DS} threshold (011) incl. crosstalk	1.25	1.5	1.75	V	(57)(58)
$V_{DS\_LS\_Th}$	Low-side V_{DS} threshold (100) incl. crosstalk	1.7	2.0	2.3	V	(57)(58)
$V_{DS\_LS\_Th}$	Low-side V_{DS} threshold (101) incl. crosstalk	2.28	2.5	2.73	V	(57)(58)
$V_{DS\_LS\_Th}$	Low-side V_{DS} threshold (110) incl. crosstalk	2.75	3.0	3.25	V	(57)(58)
$V_{DS\_LS\_Th}$	Low-side V_{DS} threshold (111) incl. crosstalk	3.23	3.5	3.78	V	(57)(58)
t_{TH_LSVDS}	Low-side V_{DS} threshold settling time	–	0.4	1.0	µs	(57)

Table 45. Low-side VDS Monitoring Electrical Specifications (continued)

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
t_{D_LSVDS}	Low-side V_{DS} comparator switching time • From LS_VDS_TH(1:0) change to V_{DS_LSTH} stable	–	0.3	1.0	µs	(57)
t_{R_LSVDS}	Low-side V_{DS} comparator recovery time after dV_D/dt • Recovery time after dV_D/dt is removed	–	–	300	ns	(57)

Note

57. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.
58. To limit the amplitude of the crosstalk to the specified value, the sequence of monitor threshold switching must not exceed a certain number of switches in any given time window. Any sequence of n commands which increase a threshold of a low-side VDS or high-side VSRC monitor must be spread over a time window t_{N_I} which meets the condition: t_{N_I} >= (n - 7) * 1.67 µs. Any sequence of m commands which decrease a threshold must be spread over a time window t_{M_D} which meets the condition: t_{M_D} >= (M - 7) * 1.67 µs.

6.5.3 Load Biasing Structures

To enable electrical diagnosis while the external load is not actuating the power stage, a voltage biasing V_{BIAS} should be applied to the load, during the idle phases.

This V_{BIAS} voltage is generated by:

- the activation of each pull-up voltage source SRC_{PUX} connected to each of the S_HSx pins. Each pull-up voltage source is supplied from VCC5.
- the activation of each pull-down current sources SRC_{PDX} connected to each of the D_LSx pins. Each pull-down voltage source referenced to ground.

When the battery voltage V_{BATT} is in the nominal range or greater, the external load is biased at a minimum voltage of typically 3.8 V. In a low battery voltage condition (V_{BATT} < 8.0 V), the load is biased at half the V_{BATT} voltage, to guarantee symmetrical voltage margins to high-side and low-side VDS comparators.

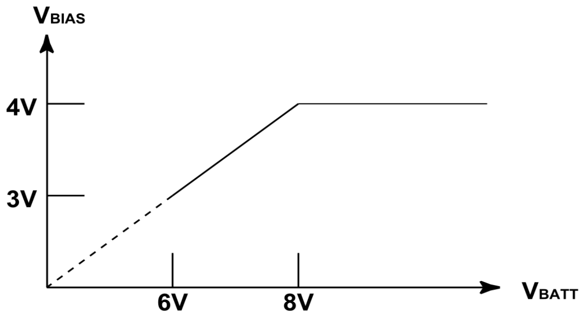


Figure 19. Biasing Voltage vs. V_{BATT}

If there is no load connected to the S_HSX pin, the voltage at this node can raise higher than V_{CC5}, due to leakage currents from V_{BOOST} and V_{BATT}.

All pull-up and pull-down structures could be switched on or off independently, under the control of the digital microcores, using the control signals hsx_bias, hsx_bias_strong, and lsx_bias. All the biasings can be enabled by the microcores using the *bias* instruction.

To prevent overloading on V_{CC5}, switching on all the high-side pull-up structures simultaneously is not possible.

Table 46. Load Biasing HS2 and HS4 Control Table

hsx_bias	hsx_bias_strong	Current Limitation (mA)	
		Min value	Max value
0	0	Biasing off	Biasing off
1	0	2.8	5.2
0	1	4.2	7.8
1	1	7.0	13.0

The pull-up voltage sources are switched off automatically, as soon as a V_{CC5} voltage is exceeded at the S_HSX pin. The voltage divider to generate the $V_{BATT}/2$ reference is disconnected from the VBATT pin as soon as RSTB is activated.

The load biasing sources can be used standalone, without using the associated low-side or high pre-driver. The D_LSX and S_HSX outputs can be connected to any node within the pin's maximum rating voltage range.

The pull-down current sources at D_LSx can also be used to slowly charge the bootstrap capacitors, after a key ON via the bootstrap path, without switching on the low-side MOSFETs. In this case, the corresponding pull-up voltage sources must be disabled.

Table 47. Load Biasing Electrical Specifications

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
I_{BIAS_HS}	Current source S_HSx (x = 1... 5) source current limitation	2.8	4.0	5.2	mA	
$I_{BIAS\_HS\_STRONG}$	Current source S_HSx (x = 2, 4) source current limitation	4.2	6.0	7.8	mA	
$I_{BIAS\_HS\_MAX}$	Total maximum current source S_HSx source current • Maximum current from VCC5	26	–	–	mA	
I_{BIAS_LS}	Current source D_LSx (x = 1...6) sink saturation current	0.98	1.09	1.2	mA	
V_{BIAS_HS}	S_HSx bias voltage regulation • VBATT > 8.0 V, VCC5 > 4.75 V • VBATT < 8.0 V, VCC5 > 4.75 V	3.8 ($V_{BATT}/2$) - 200 mV	– $V_{BATT}/2$	V_{CC5} ($V_{BATT}/2$) + 200 mV	V	(60)(60)
$V_{S\_HS\_BIAS}$	S_HSx voltage range when load biasing is switched on (S_HSx current source is switched off automatically when S_HSx is above 5.0 V) • Transients t < 400 ns • Transients t < 800 ns	-3.0 -8.0 -6.0	– – –	72 – –	V	(60)
$V_{D\_LSX\_BIAS}$	D_LSx voltage range when load biasing is switched on • Transients t < 400 ns	-3.0 -8.0	– –	75 –	V	(60)
ΔV_{BIAS_HS}	Voltage dropout across HS1/3/5 current source • VCC5 = 4.75 V, IBIAS_HS1/3/5 = 2.8 mA, IBIAS_HS_Max = 18.4 mA	–	–	0.95	V	
ΔV_{BIAS_HS}	Voltage dropout across HS2/4 current source • VCC5 = 4.75 V, IBIAS_HS2/4 = 7.0 mA, IBias_HS_Max = 18.4 mA	–	–	0.95	V	
ΔV_{BIAS_LS}	Voltage dropout across LS current source • IBIAS_LS = saturation current • IBIAS_LS = 500 µA • IBIAS_LS = 300 µA	– – –	– – –	2.5 700 400	V mV mV	
R_{BIAS_LS}	Equivalent resistance of LS current source • VD_LSx < 1.0 V	0.5	–	1.5	kΩ	

Table 47. Load Biasing Electrical Specifications (continued)

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
C_{S_HSX}	S_HSx capacitive load to GND connected via, L = 2.0 nH… 200 nH and R = 2.0 mΩ… 200 mΩ	0.01	–	25	nF	(60)

Note

59. The V_{BIAS_HS} value is specified with a load series resistor load and the corresponding low-side and high-side load biasing turned on.
60. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.6 Current Measurement

Four current measurement blocks are implemented into the 33816:

- Three general purpose blocks
- One extended mode block for DC-DC Converters

6.6.1 General Purpose Current Measurement Block

The actuator current flowing in an external sense resistor is measured to implement a closed loop current control. The current measurement block is comprised of a differential amplifier, sensing the voltage across the sense resistor, a voltage comparator, and an 8-bit current DAC.

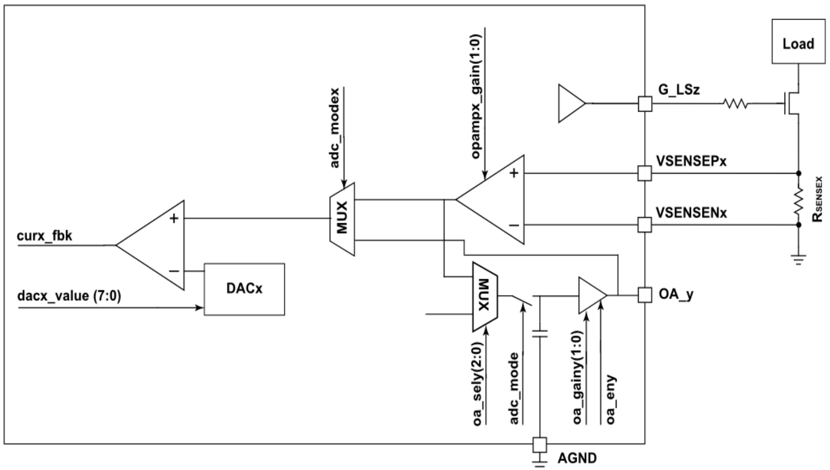


Figure 20. General Purpose Current Measurement Block Diagram

The differential amplifier gain is selectable among four different values by means of the opamapx_gain(1:0) signal, to get the suitable signal amplification. The gain can be changed at runtime by the microcore.

The differential amplifier also adds a constant offset to its output. Therefore, the output of the amplifier is always positive.

The desired actuator current level can be selected and changed at runtime by the microcore, setting the proper threshold value dacx_Value (7:0) in the DAC. Each current measurement channel can be used in ADC mode. A track and hold circuit is implemented to keep the voltage at the comparator input stable during the ADC conversion.

The differential amplifier output can be routed to an external pin (OA_1 and OA_2). In this configuration, the device output is usually connected to an ADC input of the MCU for safety and test purposes. The output multiplexer block contains an output amplifier with selectable gain by means of the oa_gainy(1:0) signal, providing full swing output on OA_y for A/D conversion, if used with 3.3 V or 5.0 V applications.

All the analog blocks for current measurement are supplied by the V_{CC5} power supply and referenced to the analog ground, AGND.

Table 48. Overall Current Sense Performance for Positive Current Measurement

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted.

Symbol	Characteristic	Statistically Evaluated	Unit	Notes
e_{CS}	Overall current sense error including gain errors and offsets at DAC range of 75%-100%, after analog offset compensation.		%	(61)
	• at GDA_diff(00) = 5.79	±3.5		
	• at GDA_diff(01) = 8.68	±3.5		
	• at GDA_diff(10) = 12.53	±3.6		
	• at GDA_diff(11) = 19.25	±3.7		
	at DAC range of 25%-75%, after analog offset compensation.			
	• at GDA_diff(00) = 5.79	±5.3		
	• at GDA_diff(01) = 8.68	±5.4		
	• at GDA_diff(10) = 12.53	±5.4		
	• at GDA_diff(11) = 19.25	±5.4		

Note

61. All input tolerances from the device specification are assumed at 6.0 σ.

6.6.2 Current Sense Amplifier

The current sense amplifier provides a voltage as detailed by the following:

- $V_{DA\_SENSE} = (V_{VSENSEPx} - V_{VSENSENx}) * G_{DA\_DIFF} + V_{DA\_BIAS}$

V_{DA_BIAS} is the fixed voltage biasing applied to the differential amplifier output.

The G_{DA_DIFF} gain value is configurable at runtime (opampx_gain(1:0)).

The allowed differential mode input voltages depend on the chosen gain value.

- $V_{DA\_DIFF\_IN} = (V_{VSENSEPx} - V_{VSENSENx})$

Table 49. Current Sense Amplifier Overall Gain Selection Table

Opampx_ gain(1:0)	Gain value	Normal differential mode typical input voltage range (mV)	Full scale current range with 10 mOhm Shunt (A)	Typical DAC resolution with 10 mOhm Shunt (mA)
00	5.79	-25.9 to 387	-2.59 to 38.7	169
01	8.68	-17.3 to 258	-1.73 to 25.8	113
10	12.53	-12.0 to 179	-1.20 to 17.9	78
11	19.25	-7.8 to 116	-0.78 to 11.6	51

The amplifier can achieve even lower voltages than V_{DA_BIAS}, when the differential input voltage is below zero, to make it able to measure small negative currents. The amplifier is fully operational down to an output voltage of typically 100 mV.

The amplifier is not designed to be used with series resistors between shunt and VSENSEPx/VSENSENx inputs.

Detection delay including comparators, is typically 50 to 500 ns, depending on gain setting, set point value, and input voltage slew rate.

Table 50. Differential Amplifiers 1, 2, 3, 4H, and 4L Electrical Specifications

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$V_{VSENSENX\_DA}$	Differential amplifier x functional range $V_{SENSENX}$ (x = 1, 2, 3, 4H, 4L) At $V_{SENSENX}$/P voltage below -1.0 V, the differential amplifier for positive current measurement can produce an output voltage > 0.0 V, even if the differential voltage at the input is < 0.0 V.	-1.0	–	1.0	V	(62)
$V_{VSENSEPX\_DA}$	Differential amplifier x functional range $V_{SENSEPX}$ (x = 1, 2, 3, 4H, 4L) At V_{SENSEN}/P voltage of below -1.0 V the differential amplifier for positive current measurement can produce an output voltage > 0.0 V even if the differential voltage at the input is < 0.0 V.	-1.0	–	1.5	V	(62)
$V_{DA\_DIFF\_IN}$	Differential input voltage range (00) • $G_{DA\_DIFF}(00) = 5.79$	-25.9	–	387	mV	(62)
$V_{DA\_DIFF\_IN}$	Differential input voltage range (01) • $G_{DA\_DIFF}(01) = 8.68$	-17.3	–	258	mV	(62)
$V_{DA\_DIFF\_IN}$	Differential input voltage range (10) • $G_{DA\_DIFF}(10) = 12.53$	-12	–	179	mV	(62)
$V_{DA\_DIFF\_IN}$	Differential input voltage range (11) • $G_{DA\_DIFF}(11) = 19.25$	-7.8	–	116	mV	(62)
$G_{DA\_DIFF}(00)$	Differential voltage gain (00)	5.71	5.79	5.87		
$G_{DA\_DIFF}(01)$	Differential voltage gain (01)	8.55	8.68	8.81		
$G_{DA\_DIFF}(10)$	Differential voltage gain (10)	12.32	12.53	12.74		
$G_{DA\_DIFF}(11)$	Differential voltage gain (11)	18.92	19.25	19.58		
$t_{DA\_GAIN\_SW}$	Gain switching settling time	–	–	2.0	µs	(62)
$SR_{DA\_DIFF\_IN}$	Differential input voltage maximum slew rate	140	–	–	mV/µs	(62)(63)
$R_{VSENSENX\_IN}$	Input impedance $V_{SENSENX}$ (x = 1, 2, 3) • 1.0 V common mode voltage	18	–	36	kΩ	
$R_{VSENSEPX\_IN}$	Input impedance $V_{SENSEPX}$ (x = 1, 2, 3) • 1.0 V common mode voltage	18	–	36	kΩ	
V_{DA_BIAS}	Output bias voltage	240	250	265	mV	
$V_{DA\_OUT\_OFF}$	Maximum output offset voltage error at maximum gain; Including amplifier input offset and bias voltage offset. Calculated using the highest gain of 19.25	-140	–	220	mV	.
V_{DA_OUT}	Differential amplifier x output voltage range	0.1	–	2.7	V	
SR_{DA}	Differential amplifier x output slew rate	2.8	–	–	V/µs	(62)

Note

62. This parameter is derived mainly from simulation and is guarantee by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.
63. Considering an external output capacitor C_{VSENSE} typically value of 330 nF. This external capacitor is recommended for improve EMI performances.

6.6.3 Current Sense DAC

In order to select the proper threshold for current control, an 8-bit current DAC is implemented to provide a threshold to the voltage comparator (dacy_Value (7:0)).

The current threshold can be calculated using the following formula.

- $I = (D_{AC\_VALUE} * V_{DAC\ LSB} - V_{DA\_BIAS})/(G_{DA\_DIFF} * R_{SENSEx})$

The D_{AC_VALUE} is selected and changed at runtime by the digital microcore by means of the signal dacx_value (7:0). A D_{AC_VALUE} below the hexadecimal value 0x0A, must be avoided, as the current sense differential amplifier does not operate with full performance at output voltages below 100 mV.

$V_{DAC\ LSB}$ is the DAC resolution.

V_{DA_BIAS} is the fixed voltage biasing applied to the differential amplifier output.

The Gain Value G_{DA_DIFF} is configurable at runtime (opampx_gain(1:0)).

R_{SENSEx} is the external sense resistor of the current measurement channel x.

Table 51. Current Sense DAC Values Examples

DAC value (hex)	DAC value (dec)	DACoutput voltage (mV)	Current threshold through 10 mOhm Shunt (A)			
			Differential voltage gain code = 00	Differential voltage gain code = 01	Differential voltage gain code = 10	Differential voltage gain code = 11
0A	10	98	-2.63	-1.76	-1.22	-0.79
…	…	…	…	…	…	…
0F	15	146	-1.79	-1.19	-0.83	-0.54
…	…	…	…	…	…	…
19	25	244	-0.1	-0.07	-0.05	-0.03
1A	26	254	0.07	0.05	0.03	0.02
1B	27	264	0.24	0.16	0.11	0.07
1C	28	273	0.4	0.27	0.19	0.12
1D	29	283	0.57	0.38	0.26	0.17
1E	30	293	0.74	0.5	0.34	0.22
1F	31	303	0.91	0.61	0.42	0.27
20	32	313	1.08	0.72	0.5	0.32
21	33	322	1.25	0.83	0.58	0.38
22	34	332	1.42	0.95	0.65	0.43
23	35	342	1.59	1.06	0.73	0.48
…	…	…	…	…	…	…
32	50	488	4.12	2.75	1.9	1.24
…	…	…	…	…	…	…
64	100	977	12.55	8.37	5.8	3.77
…	…	…	…	…	…	…
96	150	1465	20.98	14	9.7	6.31
…	…	…	…	…	…	…
FF	255	2490	38.69	25.81	17.88	11.64

Table 52. DAC 1, 2, 2, 4L and 4H Electrical Specifications

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
V_{DAC_LSB}	DAC LSB	–	9.77	–	mV	
$V_{DAC\_OUT\_MIN}$	DAC minimum output voltage • DAC code = 0x00	–	0.0	–	V	
$V_{DAC\_OUT\_MAX}$	DAC maximum output voltage • DAC code = 0xFF	–	2.49	–	V	
E_{DAC_GAIN}	DAC maximum gain error; Error of bandgap reference voltage	-1.0	–	1.0	%	
E_{DAC_DNL}	DAC differential linearity error	-0.5	–	0.5	LSB	
E_{DAC_INL}	DAC integral linearity error	-1.0	–	1.0	LSB	
$V_{DAC\_OUT\_OFF}$	DAC maximum output offset	0.0	–	10	mV	
t_{DAC}	DAC settling time	–	–	0.9	µs	(64)

Note

64. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.6.4 Current Sense Comparator

The voltage comparator toggles when the differential amplifier output exceeds the threshold provided by the DAC. The comparator output is high if the differential amplifier output is greater than the DAC output. No hysteresis is implemented. The curx_fbk comparator output is directly acquired by the digital microcore.

Table 53. Comparator Electrical Specifications

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
V_{COMP_IN}	Comparator input voltage	0.0	–	2.7	V	(65)
$V_{COMP\_IN\_OFF}$	Comparator input offset voltage	-25	–	10	mV	

Note

65. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

The detection delay from current threshold, reached to the curx_fbk comparator output toggling, is provided in Table 54.

Table 54. Current Measurement Channel 1, 2, 3, and 4H and 4L Detection Delays Specifications

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
t_{D_CS}	Detection delay coming from differential amplifier and comparator at GDA_DIFF(00) = 5.79	20	–	500	ns	
$t_{D\_CS\_150\_SLOW}$	Detection delay coming from differential amplifier and comparator for set point value 150 mV and input voltage slew rate of 2.0 mV/µs • G_{DA_DIFF}(00) = 5.79 • G_{DA_DIFF}(01) = 8.68 • G_{DA_DIFF}(10) = 12.53 • G_{DA_DIFF}(11) = 19.25	80 100 130 140	– – – –	260 270 320 400	ns	(66)
$t_{D\_CS\_SLOW}$	Detection delay coming from differential amplifier and comparator for set point value 400 mV to 2.35 V and input voltage slew rate of 2.0 mV/µs • G_{DA_DIFF}(00) = 5.79 • G_{DA_DIFF}(01) = 8.68 • G_{DA_DIFF}(10) = 12.53 • G_{DA_DIFF}(11) = 19.25	45 90 110 140	– – – –	260 280 300 350	ns	(66)
$t_{D\_CS\_150\_MID}$	Detection delay coming from differential amplifier and comparator for set point value 150 mV and input voltage slew rate of 20 mV/µs • G_{DA_DIFF}(00) = 5.79 • G_{DA_DIFF}(01) = 8.68 • G_{DA_DIFF}(10) = 12.53 • G_{DA_DIFF}(11) = 19.25	80 85 100 120	– – – –	200 220 260 300	ns	(66)
$t_{D\_CS\_MID}$	Detection delay coming from differential amplifier and comparator for set point value 400 mV to 2.35 V and input voltage slew rate of 20 mV/µs • G_{DA_DIFF}(00) = 5.79 • G_{DA_DIFF}(01) = 8.68 • G_{DA_DIFF}(10) = 12.53 • G_{DA_DIFF}(11) = 19.25	55 50 80 100	– – – –	170 180 200 240	ns	(66)
$t_{D\_CS\_150\_FAST}$	Detection delay coming from differential amplifier and comparator for set point value 150 mV and input voltage slew rate of 140 mV/µs • G_{DA_DIFF}(00) = 5.79 • G_{DA_DIFF}(01) = 8.68 • G_{DA_DIFF}(10) = 12.53 • G_{DA_DIFF}(11) = 19.25	60 70 80 100	– – – –	160 180 220 280	ns	(66)
$t_{D\_CS\_FAST}$	Detection delay coming from differential amplifier and comparator for set point value 400 mV to 2.35 V and input voltage slew rate of 140 mV/µs • G_{DA_DIFF}(00) = 5.79 • G_{DA_DIFF}(01) = 8.68 • G_{DA_DIFF}(10) = 12.53 • G_{DA_DIFF}(11) = 19.25	50 55 80 100	– – – –	115 130 160 200	ns	(66)
$t_{D\_CS\_SLOW}$	Detection delay transition 0 to 1 coming from differential amplifier and comparator for input voltage slew rate of 20 mV/µs • G_{DA_DIFF}(00) = 5.79 and set point value of 300 mV to 2.35 V • G_{DA_DIFF}(01) = 8.68 and set point value of 300 mV to 2.35 V • G_{DA_DIFF}(10) = 12.53 and set point value of 300 mV to 2.35 V • G_{DA_DIFF}(11) = 19.25 and set point value of 300 mV to 2.35 V	70 55 80 110	– – – –	140 150 190 250	ns	(66)

Table 54. Current Measurement Channel 1, 2, 3, and 4H and 4L Detection Delays Specifications (continued)

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$t_{D\_CS\_SLOW}$	Detection delay transition 1 to 0 coming from differential amplifier and comparator for input voltage slew rate of 20 mV/µs • G_{DA_DIFF}(00) = 5.79 and set point value of 300 mV to 2.35 V • G_{DA_DIFF}(01) = 8.68 and set point value of 300 mV to 2.35 V • G_{DA_DIFF}(10) = 12.53 and set point value of 300 mV to 2.35 V • G_{DA_DIFF}(11) = 19.25 and set point value of 300 mV to 2.35 V	 60 85 90 120	 – – – –	 180 200 220 270	ns	(66)
$t_{D\_CS\_FAST}$	Detection delay transition 0 to 1 coming from differential amplifier and comparator for input voltage slew rate of 140 mV/µs • G_{DA_DIFF}(00) = 5.79 and set point value of 300 mV to 2.35 V • G_{DA_DIFF}(01) = 8.68 and set point value of 300 mV to 2.35 V • G_{DA_DIFF}(10) = 12.53 and set point value of 300 mV to 2.35 V • G_{DA_DIFF}(11) = 19.25 and set point value of 300 mV to 2.35 V	 50 55 75 90	 – – – –	 120 140 160 190	ns	(66)
$t_{D\_CS\_FAST}$	Detection delay transition 1 to 0 coming from differential amplifier and comparator for input voltage slew rate of 140 mV/µs • G_{DA_DIFF}(00) = 5.79 and set point value of 300 mV to 2.35 V • G_{DA_DIFF}(01) = 8.68 and set point value of 300 mV to 2.35 V • G_{DA_DIFF}(10) = 12.53 and set point value of 300 mV to 2.35 V • G_{DA_DIFF}(11) = 19.25 and set point value of 300 mV to 2.35 V	 55 65 75 95	 – – – –	 130 150 170 210	ns	(66)

Note

66. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.6.5 Current Measurement Offset Compensation

An analog offset compensation balances the input offset of the current measurement amplifiers 1 to 4. The offset compensation enablement is managed by the digital microcores. The offset compensation must be calibrated while there is no current flowing through the sense resistor of the related measurement channel.

To perform the offset compensation, the DAC output voltage must be set to the bias voltage corresponding to the digital value 0x1A. This DAC is automatically set-up when the offset compensation is started by the digital core, using the *stoc* instruction. At the end of the offset compensation sequence, the curx_fbk comparator output signal is always low.

Each new offset compensation starts, based on the result of the previous offset compensation run, for this current measurement channel. If the offset compensation is stopped from the digital microcore while the analog offset compensation is not finished, the procedure is aborted, maintaining the last compensation value reached when the procedure was interrupted.

A residual output offset, after offset compensation completion V_{OFFDAC_LSB}, can remain for the path via the comparator to the feedback signal, and $V_{CS\_OAX\_OFF}$ when using the path to the OA_x amplifier input.

The offset compensation should be done for the maximum differential amplifier gain value used in the application. The offset compensation must be performed when a large device temperature change is expected, due to a temperature drift of the differential amplifier input offset.

Table 55. Differential Amplifier 1, 2, 3, 4L, and 4H Analog Offset Compensation Electrical Specifications

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$V_{OFFDAC\_OUT\_MAX\_POS}$	Offset compensation voltage range referred to amplifier output offset at maximum gain • $G_{DA\_DIFF}(11)$ = 19.25, Offset DAC value = +31	150	–	310	mV	
$V_{OFFDAC\_OUT\_MAX\_NEGT}$	Offset compensation voltage range referred to amplifier output offset at maximum gain • $G_{DA\_DIFF}(11)$ = 19.25, Offset DAC value = -31	-310	–	-150	mV	
V_{OFFDAC_LSB}	Offset compensation step size referred to amplifier output offset at maximum gain • $G_{DA\_DIFF}(11)$ = 19.25	5.0	–	10	mV	
$V_{OFFCOMP\_RES}$	Offset compensation digital result	-31	–	31		
V_{CS_OFF}	Residual offset after offset compensation at differential amplifier output for path shunt to comparator output. Assuming a zero DAC gain error and INL.	-0.61 -6.1	– –	0.39 3.9	LSB mV	(67)
$V_{CS\_OFF\_TEMP}$	Differential amplifier output offset temperature drift • -40 °C ≤ T_J ≤ 150 °C	-5.0 -50	– –	5.0 50	LSB mV	(67)
$V_{CS\_OAX\_OFF}$	Residual offset after offset compensation at differential amplifier output for path shunt to OAx amplifier input This includes the offset of the DAC and comparator. For the path to the OAx amplifier, these offsets are not compensated	-28.6	–	36.4	mV	(67).
$V_{DAC\_OUT\_COMP}$	DAC output voltage to perform offset compensation • DAC code = 0x1A	–	253.9	–	mV	(67)
$t_{OFFCOMP\_STEP}$	Offset compensation minimum step time • $G_{DA\_DIFF}(11)$ = 19.25	–	–	2.0	µs	
$t_{OFFCOMP}$	Offset compensation runtime to finish compensation • $G_{DA\_DIFF}(11)$ = 19.25	–	–	62	µs	(67)

Note

67. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.7 Current Measurement for DC-DC Conversion

The inputs of the 4th current sense need to support a wide range of applications. Typical applications use the 4th current sense block, either identically to the other current sense blocks or to control a DC-DC converter with a low-side current measurement, and concurrently provide an overcurrent supervision at the booster capacitor.

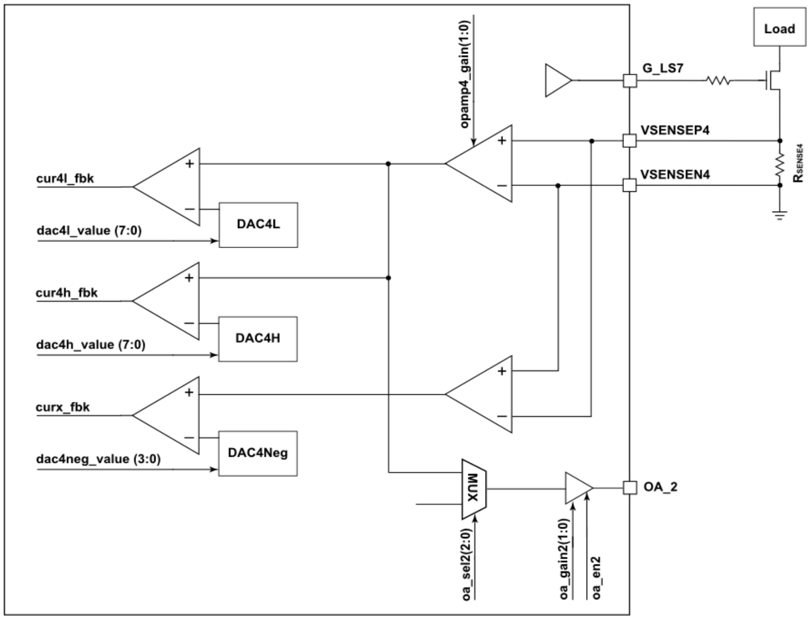


Figure 21. 4th Current Measurement Block Diagram

The two-point current control of a DC-DC converter is optimized, such as to reach a low latency of the control loop. This architecture is able to provide a short delay from the VSENSE inputs to the G_LS7 output.

The digital core contains hard wired logic for a two-point current regulation, using the cur4h_fbk and cur4l_fbk signals as inputs, and directly driving to output the input of the G_LS7 low-side driver.

A third comparator is implemented to detect negative current into the R_{SENSE} sense resistor.

A $V_{SENSEN4/P4}$ voltage of below -1.0 V, the differential amplifier of channel four for positive current measurement, can produce an output voltage > 0.0 V, even if the differential voltage at the input is < 0.0 V. This could lead to false information at the comparator output and has to be considered in the application.

The two-point current controls are made of two parallel circuitries (the sub-channels 4h and 4L) having the electrical characteristics of the single topology implemented into the three other current measurement blocks.

6.7.1 Negative Current Differential Amplifier

The Diff Ampl 4 Negative works at negative differential input voltages and therefore has a negative gain. If positive differential input voltages are applied, the amplifier output behavior is monotonic.

Table 56. Overall Current Sense Performance for Negative Current Measurement

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted.

Symbol	Characteristic	Statistically Evaluated	Unit	Notes
e_{CS_NEG}	Overall current sense error including gain errors and offsets at DAC range of 75%-100% • at GDAneg_diff=-2.0 at DAC range of 25%-75% • at GDAneg_diff=-2.0	 ±4.4 ±8.9	%	(68)

Note

68. All input tolerances from the device specification are assumed at 6.0 σ.

Table 57. Differential Amplifier 4 Negative

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$V_{VSENSEN4\_DANEG}$	Differential amplifier,4 negative (negative currents), functional range VSenseN4	-3.0	–	1.0	V	(69)
$V_{VSENSEP4\_DANEG}$	Differential amplifier, 4 negative (negative currents), functional range VSenseP4	-4.2	–	1.0	V	(69)
$V_{DANEG\_DIFF\_IN}$	Differential input voltage range • G_{DANEG_DIFF} = -2.0	-1.125	–	0.0	V	(69)
G_{DANEG_DIFF}	Differential voltage gain	-1.966	-2.0	-2.034		
$SR_{DANEG\_DIFF\_IN}$	Differential input voltage maximum slew rate	140	–	–	mV/µs	(69)
$R_{VSENSEN4\_IN}$	Input impedance VsenseN4 • 1.0 V common mode voltage	12	–	21	kΩ	
$R_{VSENSEP4\_IN}$	Input impedance VsenseP4 • 1.0 V common mode voltage	12	–	21	kΩ	
$V_{DANEG\_IN\_OFF}$	Differential amplifier maximum input offset voltage	-20	–	20	mV	
V_{DANEG_BIAS}	Output bias voltage	240	250	265	mV	
$V_{DANEG\_OUT\_OFF}$	Maximum output offset voltage error, including amplifier input offset and bias voltage offset.	-60	–	60	mV	
V_{DANEG_OUT}	Differential amplifier x output voltage range	0.0	–	2.7	V	(69)
SR_{DANEG}	Differential amplifier x output slew rate	0.28	–	–	V/µs	(69)

Note

69. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.7.2 Negative Current DAC

The DAC implemented for the negative current detection of conversion is a 4-bit DAC. The current threshold can be calculated using the following formula.

- $I = (D_{ACNEG\_VALUE} * V_{DACNEG\_LSB} - V_{DANEG\_BIAS})/(G_{DANEG\_DIFF} * R_{SENSE4})$

D_{ACNEG_VALUE} is selected and changed at runtime by the digital microcore (dac4neg_values(3:0))

V_{DACNEG_LSB} is the DAC resolution.

V_{DANEG_BIAS} is the fixed voltage biasing applied to the differential amplifier output.

G_{DANEG_DIFF} is the amplifier gain of the negative current measurement stage.

R_{SENSE4} is the external sense resistor of the current measurement channel 4.

Table 58. Negative Current Sense DAC Values Examples

DAC neg value (hex)	DAC neg value (dec)	DAC neg output voltage (mV)	Current Threshold through 10 mOhm Shunt (A)
2	2	313	-3.13
3	3	469	-10.94
4	4	625	-18.75
5	5	781	-26.56
6	6	938	-34.38
7	7	1094	-42.19
8	8	1250	-50
9	9	1406	-57.81
A	10	1563	-65.63
B	11	1719	-73.44
C	12	1875	-81.25
D	13	2031	-89.06
E	14	2188	-96.88
F	15	2344	-104.69

Table 59. DAC 4 Neg Electrical Characteristics

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
V_{DACNEG_LSB}	DAC LSB	–	156.3	–	mV	
$V_{DACNEG\_OUT\_MIN}$	DAC minimum output voltage • DAC code = 0x0	–	0.0	–	V	
$V_{DACNEG\_OUT\_MAX}$	DAC maximum output voltage • DAC code = 0xF	–	2.344	–	V	
E_{DACNEG_GAIN}	DAC maximum gain error	-1.0	–	1.0	%	
E_{DACNEG_DNL}	DAC differential linearity error	-0.063	–	0.063	LSB	
E_{DACNEG_INL}	DAC integral linearity error	-0.063	–	0.063	LSB	
$V_{DACNEG\_OUT\_OFF}$	DAC maximum output offset	0.0	–	10	mV	

Table 59. DAC 4 Neg Electrical Characteristics (continued)

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
t_{DACNEG}	DAC settling time	–	–	0.9	µs	(70)

Note

70. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.7.3 Negative Current Comparator

The two positive current comparators implemented into the DC/DC current measurement block have the same behavior and characteristics of the one implemented into the three other current measurement blocks.

The voltage comparator electrical characteristics dedicated to the negative current comparator are described by the following.

Table 60. Voltage Comparator 4 Neg Electrical Characteristics

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
V_{COMP_IN}	Comparator input voltage	0.0	–	2.7	V	(71)
$V_{COMP\_IN\_OFF}$	Comparator input offset voltage	-25	–	10	mV	

Note

71. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.7.4 Negative Current Sense

The negative voltage comparator toggles when the differential amplifier output exceeds the threshold provided by the DAC. The comparator output is high if the differential amplifier output is greater than the DAC output. No hysteresis is implemented. The comparator output cur4h_fbk, cur4l_fbk, and cur4neg_fbk are directly acquired by the digital microcore.

Table 61. Current Measurement Channel 4 Neg Detection Delays

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
t_{D_CSNEG}	Detection delay coming from differential amplifier and comparator at G_{DANEG_DIFF} = -2.0	80	–	500	ns	
$t_{D\_CSNEG\_150\_MID}$	Detection delay coming from differential amplifier and comparator for set point value 150 mV and input voltage slew rate of 20 mV/µs • G_{DANEG_DIFF} = -2.0	80	–	200	ns	(72)
$t_{D\_CSNEG\_MID}$	Detection delay coming from differential amplifier and comparator for set point value 400 mV to 2.35 V and input voltage slew rate of 20 mV/µs • G_{DANEG_DIFF} = -2.0	75	–	160	ns	(72)
$t_{D\_CSNEG\_150\_FAST}$	Detection delay coming from differential amplifier and comparator for set point value 150 mV and input voltage slew rate of 140 mV/µs • G_{DANEG_DIFF} = -2.0	55	–	160	ns	(72)

Table 61. Current Measurement Channel 4 Neg Detection Delays (continued)

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$t_{D\_CSNEG\_FAST}$	Detection delay coming from differential amplifier and comparator for set point value 400 mV to 2.35 V and input voltage slew rate of 140 mV/µs • G_{DANEG_DIFF} = -2.0	50	–	120	ns	(72)

Note

72. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.7.5 Current Measurement Offset Compensation

There is no analog offset compensation for the differential amplifier 4 negative.

6.8 OA_x Output Pin and Multiplexer

6.8.1 General Features

The output signals of the four current sensing amplifiers are available via two external pins of the device, OA_1 and OA_2. A oa_gainx(1:0) gain can be applied to this output signal, such as to rescale the signal and adapt it to an ADC input range of 3.3 or 5.0 V.

Four values are available for the oa_gainx(1:0) gain. The maximum output voltage at the OA_x pins depends on the V_{CC5} value.

Table 62. OA_x Amplifier Gain Selection and Output Voltage

OAGainx(1:0)	Gain value	Output voltage
00	1.33	V_{IN} * OAG_{AINX}
01	2.0	V_{IN} * OAG_{AINX}
10	3.0	(V_{IN} -250 mV) * OAG_{AINX} + 250 mV
11	5.33	(V_{IN} -250 mV) * OAG_{AINX} + 250 mV

For the two higher gains of 3.0 and 5.33, the bias voltage of nominal 250 mV of the input signal is removed before amplifying the signal, then added to the amplified signal.

The OA_1 and OA_2 output pins include the possibility of switching to high-impedance mode. This feature allows connection these pins to the same MCU ADC input and perform sequential conversions. Both OA_x output multiplexers can be optionally switched to VCC2P5.This feature allows checking the connection of the MCU ADC input pin. The OA_x multiplexers output configuration is managed by the digital cores. The configuration tables are given in Table 63 and Table 64.

Table 63. OA_1 Multiplexer Logic Table(73)

OaSel1(2:0)	OaEN1	Signal at output OA_1
000	1	OA_Cur 1 (Feedback of current measurement 1)
001	1	OA_Cur 3 (Feedback of current measurement 3)
010	1	Reserved
011	1	Reserved
100	1	Reserved
101	1	VCC2P5
110	1	Reserved

Table 63. OA_1 Multiplexer Logic Table[73] (continued)

111	1	Reserved
xxx	0	HiZ - High-impedance

Notes

73. The current measurement 1 and 3 feedbacks can only be routed to the output OA_1. The current measurement 2 and 4 feedbacks can only be routed to the output OA_2.

Table 64. OA_2 Multiplexer Logic Table [74]

OaSel2(2:0)	OaEN2	Signal at output OA_2
000	1	OA_Cur 2 (Feedback of current measurement 2)
001	1	OA_Cur 4 (Feedback of current measurement 4)
010	1	Reserved
011	1	Reserved
100	1	Reserved
101	1	VCC2P5
110	1	Reserved
111	1	Reserved
xxx	0	HiZ - High-impedance

Notes

74. The current measurement 1 and 3 feedbacks can only be routed to the output OA_1. The current measurement 2 and 4 feedbacks can only be routed to the output OA_2.

The OA_x output multiplexers are switched according to the signals OaSel1(2:0) and OaSel2(2:0). These two signal are respectively set by means of the 3-bit word oa1_source in the Oa_out1_config register (0x1AA) and the 3-bit word oa2_source in the Oa_out2_config register (0x1AA). Moreover, the output's signal gains are set in these two registers.

6.8.2 OA_x Pin Digital I/O Function

The OA_1 and OA_2 pins are configurable as digital flag bus inputs or outputs, flag(10) and flag(11). This can be selected by a SPI configuration, by means of the Flags_source (0x1C3) and Flags_direction (0x1C1) registers. As soon as the pin is configured as a digital input, the buffer is switched to high-impedance.

Table 65. OAx Enable Truth Table

flags_source	flags_direction	opamp_pin_ source(x) (reset=0)	oa_enx (reset=0)	i_o_opamp(x) (reset=0)	OA_x Buffer state	Description
0	–	0	0	–	HiZ	OA_x pin is used as an analog output, enable signal is low
0	–	0	1	–	On	OA_x pin is used as an analog output, enable signal is high
1	1	1	–	0	HiZ	OA_x pin is used as an digital input
1	0	1	–	1	On	OA_x pin is used as an digital output

6.8.3 A/D Multiplexer and OA_x Output Amplifier Enablement

The A/D multiplexer routes the analog or the digital output functionality to each OA_x pin. If the digital output function is selected, the digital flag pin is configured as an input, and the corresponding OA_x output amplifier is switched to HiZ.

Table 66. Multiplexer A/D Truth Table

opamp_pin_source (x)	Output selection
0 (reset value)	Analog output function
1	Digital output function (opamp_flag_out1/2 used)

6.8.4 OA_x Pin I/O Voltage

The I/O voltage of the OA_x pins is not automatically set according to the VCCIO voltage supplied to the device. The OA_x output amplifier is supplied by VCC5, and is also used for the digital output function. The digital input signal to the OA_x output amplifier is a VCC2P5 based signal, so the I/O voltage is selected according to the gain value of the OA_x output amplifier.

Table 67. OA_x Amplifier Gain Selection

Oagain(1:0)	Typical gain value	Used for
00 (reset value)	1.33	3.3 V I/O
01	2.0	5.0 V I/O
10	2.0	–
11	5.33	–

6.8.5 Weak Pull-down Resistor

A weak pull-down resistor is implemented on each OA_x input/output. This resistor is always present, whatever digital or analog functionality is selected.

6.8.6 OA_x Output Offset and Offset Error

The current measurement amplifier's output signal has a fixed offset of typically 250 mV, and a variable residual offset of -28.6 to +36.4 mV on the path after analog offset compensation. This offset depends on the current measurement amplifier's gain setting, and is amplified by the OA_x amplifier gain. In addition, a the OA_x amplifier adds input offset of ±10 to ±13.5 mV.

For the two higher gains of 3.0 and 5.33, the bias voltage of a nominal 250 mV of the input signal is removed before amplifying the signal and added again to the amplified signal afterwards.

Table 68 describes some examples for load currents, gain settings, and corresponding output voltage ranges. Note that this calculation only takes into account the offset errors. The other errors must be considered in a full error calculation.

Table 68. OAx Input and Output Values

Load Current at 10 mOhm Shunt (A)	Current Measurement Amplifier Gain Setting	OAx Amplifier Gain Setting	OAx Output Voltage Min (mV)	OAx Output Voltage Typ (mV)	OAx Output Voltage Max (mV)
0	8.68	1.33	281	333	394
25.8	8.68	1.33	3261	3312	3374
0	19.25	5.33	44	250	497
1.0	19.25	5.33	1070	1276	1523
2.5	19.25	5.33	2609	2815	3062

Table 69. OAx Output Pin and Multiplexer Electrical Characteristics

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
V_{OAX}	OA_x output voltage range	0.0	–	V_{CC5}	V	(75)
GBW_{OAX}	OA_x amplifier gain bandwidth product	2.5	–	7.0	MHz	(75)
BW_{OAX}	OA_x output bandwidth	100	–	–	kHz	(75)(76)
SR_{OAX}	OA_x output slew rate	2.0	–	–	V/µs	(75)
C_{OAX}	OA_x permissible capacitive load • w/o series resistor, for digital function • R_{MIN} = 50 Ohm • R_{MIN} = 75 Ohm • R_{MIN} = 100 Ohm • R_{MIN} = 200 Ohm	 – 50 15 5.0 1.0	 – – – – –	 50 100 50 15 5.0	 pF nF nF nF nF	(75)
$PSRR_{OAX}$	OA_x power supply rejection	–	–	103	dB	(75)
$G_{OAX}(00)$	OA_x output gain (00)	1.303	1.33	1.357		
$G_{OAX}(01)$	OA_x output gain (01)	1.94	2.0	2.06		
$G_{OAX}(10)$	OA_x output gain (10)	2.91	3.0	3.09		
$G_{OAX}(11)$	OA_x output gain (11)	5.17	5.33	5.49		
$G_{OAX}(ADC)$	OA_x output gain (ADC)	0.98	1.0	1.02		
t_{OAX_GAIN}	OA_x output gain switching time	–	–	2.0	µs	(75)
V_{OAX_OFFSET}	OAx output offset voltage from OA_x amplifier • G_{OAx} = 1.0 • G_{OAx} = 1.33 • G_{OAx} = 2.0 • G_{OAx} = 3.0 • G_{OAx} = 5.33	 -14 -18 -28 -30 -53	 – – – – –	 14 18 28 30 53	mV	
R_{OAX_EN0}	OA_x input impedance when OaENx = 0 • 2.0 V, impedance to GND	 350	 –	 –	kΩ	
t_{OAX_MUX}	OA_x multiplexer switching time	–	–	10.0	µs	(75)(77)
$V_{OAx\_Drift\_ADC}$	OAx output voltage drift of T&H in ADC mode over time • at VOAx=1.5 V and after 20 µs	-50	–	50	mV	

Notes

75. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.
76. In the path of the current measurement output signal from the differential amplifier to the OA_x pin there is a parasitic low pass filter limiting the bandwidth of this path.
77. When switching the OAx multiplexer of one path this can introduce a glitch on the output signal of the other output. The worst case duration of the glitch is below 10 µs. Moreover the settling time of the switched path this prolonged.

6.9 PLL and Backup Clock

The digital logic is supplied by a clock (cksys) whose operating frequency can be set to 24 MHz or 12 MHz. This selection can be achieve via a SPI configuration bit. After reset, the default operating frequency is set to 24 MHz. This clock is generated by a PLL, based on the external reference signal applied to the CLK pin. The internal PLL generates a typical 48 MHz or 24 MHz clock. Two internal clocks are derived from the PLL:

- the main logic clock cksys
- the code RAM clock cksys_cram inverted in respect to cksys
- the Data RAM clock cksys_dram inverted in respect to cksys.

If an unsuitable signal is applied on the CLK pin, the device automatically switches to the internal clock generated by an integrated backup oscillator. When a suitable signal is retrieved on the CLK pin, the MCU interfaced to the 33816 must request to switch back to the external reference clock through the SPI. The switch back to the external clock is not automatic. Around 25 μs is required to lock the PLL the first time or to re-lock it.

The PLL circuitry is supplied by the VCC5 pin. The PLL is started as soon as the supply voltages are stable and the input clock is present. The PLL works down to a VCC5 voltage of typically 4.0 V.

The RESETB pin state has no effect on the PLL.

The clock monitor detects an invalid PLL output clock, either by a missing PLL lock signal, or by supervising the output frequency of the PLL. Eight backup clock cycles are required to detect a wrong output clock frequency of the PLL. When switching from the external reference to the backup clock, it takes some additional time until the PLL is relocked. As long as the PLL output clock is not stable (PLL not locked), the signal cksys_missing is set to 1. This signal cksys_missing is used in the digital core to generate an interrupt. During a cksys_missing condition, there is the option to switch off all pre-drivers asynchronously by the cksys_drven signal. This configuration is done by setting to ‘1’ the bit cksys_miss_dis_drv of the Backup_status_clock_reg (0x1C7)

The PLL output frequency can be modulated. Modulation activation is enabled by default, but can be disabled through the SPI in the PLL_config register (0x1C6).

Table 70. PLL and Back-Up Clock Electrical Characteristics

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
f_{CLK}	CLK pin input frequency • PLL operating frequency at V_{CC2P5} > 2.0 V • PLL operating frequency at V_{CC2P5} > 2.21 V	 0.95 0.94	 1.0 1.0	 1.05 1.06	MHz	(78)
DC_{CLK}	CLK pin input duty cycle	45	50	55	%	
V_{CLK}	CLK pin voltage	0.0	–	V_{CC5}	V	(79)
V_{IH_CLK}	CLK pin high input voltage threshold	1.5	–	2.2	V	
V_{IL_CLK}	CLK pin low input voltage threshold	1.0	–	1.65	V	
V_{HYST_CLK}	CLK pin hysteresis	0.3	–	–	V	
t_{CLK_JITTER}	CLK pin clock edge jitter	-25	–	25	ns	(79)
f_{CLK_BACK}	Backup oscillator clock frequency	0.95	1.0	1.05	MHz	
DC_{CLK_BACK}	Backup oscillator clock duty cycle	48	50	52	%	
$f_{CKSYS24}$	cksys output clock frequency 24 MHz	$f_{CLK(\_BACK)}$ * 23.5	$f_{CLK(\_BACK)}$ * 24	$f_{CLK(\_BACK)}$ * 24.5	MHz	
f_{CKSYS_RAM24}	cksys_c/dram output clock frequency 24 MHz	$f_{CLK(\_BACK)}$ * 23.5	$f_{CLK(\_BACK)}$ * 24	$f_{CLK(\_BACK)}$ * 24.5	MHz	
$f_{CKSYS12}$	cksys output clock frequency 12 MHz	$f_{CLK(\_BACK)}$ * 11.75	$f_{CLK(\_BACK)}$ * 12	$f_{CLK(\_BACK)}$ * 12.25	MHz	

Table 70. PLL and Back-Up Clock Electrical Characteristics (continued)

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
f_{CKSYS_RAM12}	cksys_c/dram output clock frequency 12 MHz	$f_{CLK(\_BACK)}$ * 11.75	$f_{CLK(\_BACK)}$ * 12	$f_{CLK(\_BACK)}$ * 12.25	MHz	
$M_{F\_MOD\_RATE}$	cksys, cksys_c/dram output clock frequency modulation rate	-2.08	–	2.08	%	
f_{CKSYS_MOD}	cksys, cksys_c/dram output clock frequency modulation frequency	–	25	–	kHz	
t_{PLL_LOCK}	PLL lock time (first lock), including digital filter time	–	25	40	µs	
t_{PLL_RELOCK}	PLL lock time (re-lock), including digital filter time	–	25	40	µs	
V_{CC5_PLLMIN}	PLL and Backup Clock minimum operating input voltage	–	–	4.0	V	
$t_{PLL\_LOCK\_FILTER}$	PLL lock signal digital filter time	–	10	–	µs	
$f_{CLK\_LOSS24\_L}$	Digital clock monitor lower threshold frequency 24 MHz • Digital clock monitor threshold 127 • Digital clock monitor threshold 163	20.11 25.81	21.17 27.17	22.23 28.53	MHz	(79)
$f_{CLK\_LOSS12\_L}$	Digital clock monitor threshold frequency 12 MHz • Digital clock monitor threshold 63 • Digital clock monitor threshold 82	9.98 12.98	10.5 13.67	11.03 14.35	MHz	(79)
t_{CLK_LOSS}	Digital clock monitor detection time, digital clock monitor running on a 1.05 MHz internal clock	–	–	8.4	µs	(79)

Notes

78. The CLK pin input duty cycle minimum value is 45%, typical value is 50% and maximum value is 55%.
79. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

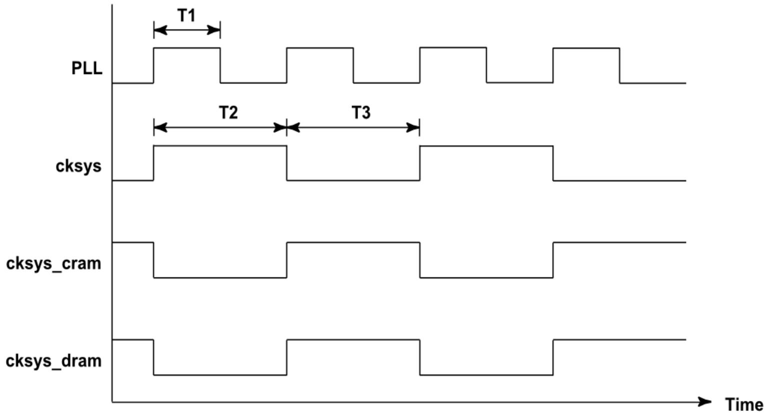


Figure 22. Timing Diagrams for cksys, cksys_cram, and cksys_dram

The following values take into account an input clock at 0.95 - 1.05 MHz, a PLL multiplication factor of 47 - 49, and an output duty cycle of 45 - 55%.

Table 71. Timing for cksys, cksys_cram, and cksys_dram

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
t_{CKSYS_T1}	cksys rising edge to cksys_cram rising edge T1, Code RAM address setup phase	8.75	–	12.32	ns	(80)
t_{CKSYS_T2}	cksys rising edge to cksys_c/dram rising edge T2, Code RAM or Data RAM address setup phase	19.44	–	–	ns	(80)
t_{CKSYS_T3}	cksys_c/dram rising edge to cksys rising edge T3, Code RAM or Data RAM address setup phase	19.44	–	–	ns	(80)

Notes

80. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.

6.10 Digital I/Os

6.10.1 General Features

The digital I/Os ensure a way of communication to the MCU. The SPI interface allows register setup, Code RAM and Data RAM download. Interfacing is made of four pins: MISO, MOSI, SCLK, and CSB. The DBG pin routes the trace code out of the device and can optionally be configured as bidirectional flags. The six STARTx pins (START1 to START6) are dedicated to the injection action trigger and can optionally be configured as bidirectional flags. The 3 FLAGx (FLAG0 to FLAG2) pin are bidirectional digital pins.The OA_x pins (OA_1 and OA_2) are analog pins that can optionally be configured as bidirectional flags.

Table 72. Digital I/Os Electrical Characteristics

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to DGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$V_{IOVCCOI}$	Digital pins voltage (IRQB, MISO, MOSI, SCLK, CSB, STARTx, FLAGx, DBG, and OAx)	0.0	–	V_{CCIO}	V	(81)
V_{IOVCC5}	Digital pins voltage (RESETB, DRVEN)	0.0	–	V_{CC5}	V	
t_{FILT_RESETB}	RESETB filter time - a pulse > 2.0 µs always causes a reset	0.2	–	2.0	µs	
V_{IH_IO}	Digital pins high input voltage threshold (RESETB, IRQB, MOSI, SCLK, CSB, DRVEN, STARTx, FLAGx, DBG, and OA_x)	1.5	–	2.2	V	
V_{IL_IO}	Digital pins low input voltage threshold (RESETB, IRQB, MOSI, SCLK, CSB, DRVEN, STARTx, FLAGx, DBG, and OA_x)	1.0	–	1.65	V	
V_{HYST_IO}	Digital pins hysteresis (RESETB, IRQB, MOSI, SCLK, CSB, DRVEN, STARTx, FLAGx, DBG, and OA_x)	0.3	–	–	V	(81)
V_{OH_XXX}	Digital pins high output voltage (IRQB, MISO, STARTx, FLAGx, and DBG) • I_{OUT} > -50 µA, no higher current at other I/Os • I_{OUT} > -1.0 mA, no higher current at other I/Os • I_{OUT} > -2.0 mA, no higher current at other I/Os	 V_{CCIO} - 0.05 V_{CCIO} - 0.3 V_{CCIO} - 0.6	 – – –	 – – –	V	
V_{OH_OAX}	Digital pins high output voltage (OA_x) • GOAx = 1.33, I_{OUT} > -50 µA • GOAx = 1.33, I_{OUT} > -1.0 mA • GOAx = 2.0, I_{OUT} > -50 µA • GOAx = 2.0, I_{OUT} > -1.0 mA	 3.15 2.8 V_{CC5} -0.15 V_{CC5} -0.6	 – – – –	 – – – –	V	(82)

Table 72. Digital I/Os Electrical Characteristics (continued)

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to DGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
V_{OL_XXX}	Digital pins low output voltage (IRQB, MISO, STARTx, FLAGx, and DBG) • I_{OUT} < 50 µA, no higher current at other I/Os • I_{OUT} < 1.0 mA, no higher current at other I/Os • I_{OUT} < 2.0 mA, no higher current at other I/Os	– – –	– – –	0.05 0.3 0.6	V	
V_{OL_OAX}	Digital pins low output voltage (OA_x) • I_{OUT} < 0.5 mA	–	–	0.3	V	(82)
t_{R_XXX}	Digital pins output rise t. (IRQB, STARTx, FLAGx, and DBG) • C_{LOAD} = 30 pF	3.0	–	12	ns	(81) (82)
t_{F_XXX}	Digital pins output fall t. (IRQB, STARTx, FLAGx, and DBG), 90%-10% of out voltage • C_{LOAD} = 30 pF	3.0	–	12	ns	(81) (82)
t_{D_XXX}	Digital pins output delay (IRQB, STARTx, FLAGx, and DBG), 10% of out voltage change • C_{LOAD} = 30 pF	2.0	–	10	ns	(81) (82)
t_{R_XXX}	Digital pins output rise t. (IRQB, STARTx, FLAGx, and DBG), 10%-90% of out voltage • C_{LOAD} = 30 pF, V_{CCIO} = 3.3 V, T_A = +50 °C • C_{LOAD} = 30 pF, V_{CCIO} = 5.0 V, T_A = +50 °C	6.0 4.0	– –	9.0 7.0	ns	(81) (82)
t_{F_XXX}	Digital pins output fall t. (IRQB, STARTx, FLAGx, and DBG), 90%-10% of out voltage • C_{LOAD} = 30 pF, V_{CCIO} = 3.3 V, T_A = +50 °C • C_{LOAD} = 30 pF, V_{CCIO} = 5.0 V, T_A = +50 °C	6.0 4.0	– –	8.0 6.0	ns	(81) (82)
t_{D_XXX}	Digital pins output delay (IRQB, STARTx, FLAGx, and DBG), 10% of out voltage change • C_{LOAD} = 30 pF, V_{CCIO} = 3.3 V, T_A = +50 °C • C_{LOAD} = 30 pF, V_{CCIO} = 5.0 V, T_A = +50 °C	4.0 3.0	– –	7.0 6.0	ns	(81) (82)
t_{DRF_XXX}	Digital pins delta between rise and fall time (IRQB, STARTx, FLAGx, and DBG), 10% of out voltage change • C_{LOAD} = 30 pF, V_{CCIO} = 3.3 V, T_A = +50 °C • C_{LOAD} = 30 pF, V_{CCIO} = 5.0 V, T_A = +50 °C	-0.5 -0.3	– –	1.6 1.0	ns	(81) (82)
t_{DD_XXX}	Digital pins delta between output delay for rising edge and falling edge (IRQB, STARTx, FLAGx, and DBG), 10% of out voltage change • C_{LOAD} = 30 pF, V_{CCIO} = 3.3 V, T_A = +50 °C • C_{LOAD} = 30 pF, V_{CCIO} = 5.0 V, T_A = +50 °C	-0.3 -0.7	– –	0.1 -0.3	ns	(81) (82)
t_{R_OAX}	Digital pins output rise time (OA_x), 10%-90% of out voltage • C_{LOAD} = 30 pF, V_{CCIO} = 3.3 V • C_{LOAD} = 30 pF, V_{CCIO} = 5.0 V	– –	– –	1.4 2.0	µs	(81) (82)
t_{F_OAX}	Digital pins output fall time (OA_x), 90%-10% of out voltage • C_{LOAD} = 30 pF, V_{CCIO} = 3.3 V • C_{LOAD} = 30 pF, V_{CCIO} = 5.0 V	– –	– –	1.4 3.2	µs	(81) (82)
t_{D_OAX}	Digital pins output delay (OA_x), 10% of out voltage change • C_{LOAD} = 30 pF, V_{CCIO} = 3.3 V • C_{LOAD} = 30 pF, V_{CCIO} = 5.0 V	– –	– –	2.7 3.0	µs	(81) (82)
CP_{IN_XXX}	Digital pins equivalent pin capacitance (IRQB, START1, START2, START3, START4, START5, START6, FLAG0, FLAG1, FLAG2, DBFG)	–	–	10	pF	(81)
CP_{IN_MISO}	Digital pins equivalent pin capacitance (MISO)	–	–	10	pF	(81)

Table 72. Digital I/Os Electrical Characteristics (continued)

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to DGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
CP_{IN_MOSI}	Digital pins equivalent pin capacitance (MOSI)	–	–	10	pF	(81)

Notes

81. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.
82. Referenced to AGND pin

6.11 SPI Interface

The device includes a SPI MISO driver with programmable slew rate control.

- The driver is optimized to have a low tolerance in rise/fall time over temperature and process.
- The rise time and fall time are auto-adapted regardless if 5.0 V or 3.3 V are supplied on the VCCIO pin.
- The options and slew rate settings are described in Table 73. Two possible slew rates can be selection by means of the bit miso_slew rate of the SPI_config register (0x1C8).

Refer to Spi_protocol Block for the SPI protocol description.

Table 73. SPI MISO Slew Rate Settings

miso_slewrate	Description	MISO Bus Load (pF)	Rise/Fall Max. Time at VCCIO = 3.3 V (ns)	Rise/Fall Max. Time at VCCIO = 5.0 V (ns)
0	Slow	30	20	18
0	Slow	75	40	40
0	Slow	150	80	70
1	Fast	30	4.9	3.6
1	Fast	75	8.4	6.3
1	Fast	150	14.8	11.3

The two slew rate setting target two different baud rate ranges:

- the fast slew rate addresses the max baud rate range of typically 8.0 to10 Mbps
- the slow slew rate addresses the baud rate range of typically 3.5 to 8.0 Mbps

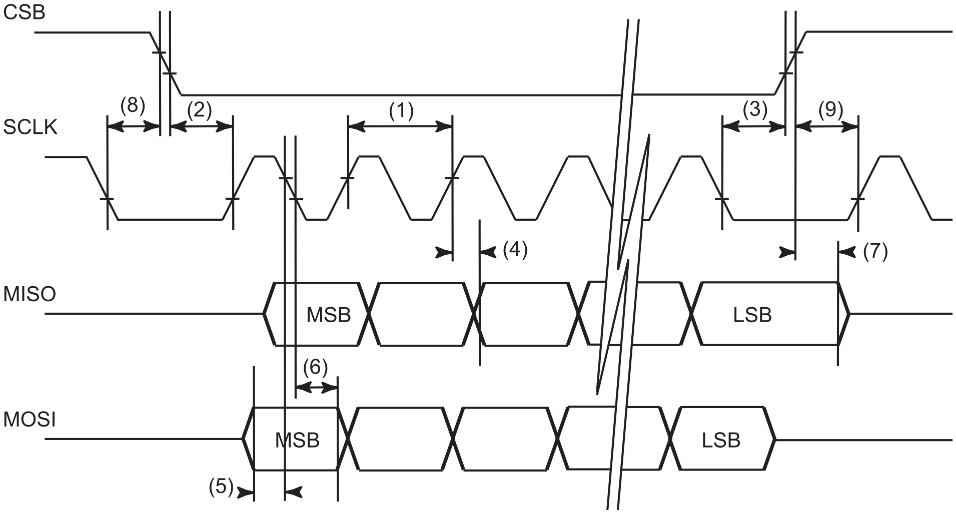


Figure 23. SPI Timing

Table 74. General SPI Electrical Characteristics

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
f_{SCLK}	SCLK pin input frequency - (1) in Figure 23	–	–	10	MHz	
t_{CSBF_SCLKR}	CSB fall to first SCLK rise - (2) in Figure 23	$1/f_{SCLK}$	–	–	ns	(83)
t_{SCLKF_CSBR}	Last SCLK fall to CSB rise - (3) in Figure 23	$1/f_{SCLK}$	–	–	ns	(83)
t_{MISO_VAL}	MISO valid time - (4) in Figure 23	–	–	$10 + t_{DR/F\_MISO}$	ns	(83)(84)
t_{MOSI_SET}	MOSI setup time - (5) in Figure 23	10	–	–	ns	(83)
t_{MOSI_HOLD}	MOSI hold time - (6) in Figure 23	12.5	–	–	ns	(83)
t_{CSBR_MISOT}	CSB rise to MISO tri-state - (7) in Figure 23	–	–	15	ns	(83)
t_{SCLKF_CSBF}	SCLK fall (other device) to CSB fall - (8) in Figure 23	13	–	–	ns	(83)
t_{CSBR_CLKR}	CSB rise to SCLK rise (other device) - (9) in Figure 23	15	–	–	ns	(83)

Notes

83. This parameter is derived mainly from simulation and is guaranteed by design characterization on a small sample size of typical devices under typical conditions, unless otherwise noted.
84. t_{DR/F_MISO} is the rise time and fall time provide inTable 75 and Table 76.

Table 75. SPI Electrical Characteristics for VCCIO = 3.3 V

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$t_{R\_MISO\_S3.3}$	MISO rise time at 10%-90% of out voltage Slow setting • C_L = 30 pF • C_L = 75 pF • C_L = 150 pF	 10 20 40	 – – –	 20 40 80	ns	
$t_{F\_MISO\_S3.3}$	MISO fall time at 90%-10% of out voltage Slow setting • C_L = 30 pF • C_L = 75 pF • C_L = 150 pF	 10 20 40	 – – –	 20 40 80	ns	
$t_{R\_MISO\_F3.3}$	MISO rise time at 10%-90% of out voltage Fast setting • C_L = 30 pF • C_L = 75 pF • C_L = 150 pF	 1.5 2.7 4.4	 – – –	 4.9 8.4 14.8	ns	
$t_{F\_MISO\_F3.3}$	MISO fall time at 90%-10% of out voltage Fast setting • C_L = 30 pF • C_L = 75 pF • C_L = 150 pF	 1.5 2.7 4.4	 – – –	 4.9 8.4 14.8	ns	
$t_{DR\_MISO\_S3.3}$	MISO pad total delay to 90% of out voltage (propagation delay plus rise time) Slow setting • C_L = 30 pF • C_L = 75 pF • C_L = 150 pF	 – – –	 – – –	 30 50 90	ns	
$t_{DF\_MISO\_S3.3}$	MISO pad total delay to 10% of out voltage (propagation delay plus fall time) Slow setting • C_L = 30 pF • C_L = 75 pF • C_L = 150 pF	 – – –	 – – –	 30 50 90	ns	
$t_{DR\_MISO\_F3.3}$	MISO pad total delay to 90% of out voltage (propagation delay plus rise time) Fast setting • C_L = 30 pF • C_L = 75 pF • C_L = 150 pF	 – – –	 – – –	 13.4 17.1 23.9	ns	
$t_{DF\_MISO\_F3.3}$	MISO pad total delay to 10% of out voltage (propagation delay plus fall time) Fast setting • C_L = 30 pF • C_L = 75 pF • C_L = 150 pF	 – – –	 – – –	 13.4 17.1 23.9	ns	

Table 76. SPI Electrical Characteristics for VCCIO = 5.0 V

Characteristics noted under conditions -40 ºC < T_A < +125 ºC, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted. Characteristics referenced to PGND pin, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
$t_{R\_MISO\_S5.0}$	MISO rise time at 10%-90% of out voltage Slow setting • C_L = 30 pF • C_L = 75 pF • C_L = 150 pF	 9.0 20 35	 – – –	 18 40 70	ns	
$t_{F\_MISO\_S5.0}$	MISO fall time at 90%-10% of out voltage Slow setting • C_L = 30 pF • C_L = 75 pF • C_L = 150 pF	 9.0 20 35	 – – –	 18 40 70	ns	
$t_{R\_MISO\_F5.0}$	MISO rise time at 10%-90% of out voltage Fast setting • C_L = 30 pF • C_L = 75 pF • C_L = 150 pF	 1.1 2.1 3.6	 – – –	 3.6 6.3 11.3	ns	
$t_{F\_MISO\_F5.0}$	MISO fall time at 90%-10% of out voltage Fast setting • C_L = 30 pF • C_L = 75 pF • C_L = 150 pF	 1.1 2.1 3.6	 – – –	 3.6 6.3 11.3	ns	
$t_{DR\_MISO\_S5.0}$	MISO pad total delay to 90% of out voltage (propagation delay plus rise time) Slow setting • C_L = 30 pF • C_L = 75 pF • C_L = 150 pF	 – – –	 – – –	 25 47 77	ns	
$t_{DF\_MISO\_S5.0}$	MISO pad total delay to 10% of out voltage (propagation delay plus fall time) Slow setting • C_L = 30 pF • C_L = 75 pF • C_L = 150 pF	 – – –	 – – –	 25 47 77	ns	
$t_{DR\_MISO\_F5.0}$	MISO pad total delay to 90% of out voltage (propagation delay plus rise time) Fast setting • C_L = 30 pF • C_L = 75 pF • C_L = 150 pF	 – – –	 – – –	 9.6 12.5 17.8	ns	
$t_{DF\_MISO\_F5.0}$	MISO pad total delay to 10% of out voltage (propagation delay plus fall time) Fast setting • C_L = 30 pF • C_L = 75 pF • C_L = 150 pF	 – – –	 – – –	 9.6 12.5 17.8	ns	

6.12 Internal Pull-up and Pull-down

The 33816 provides internal pull-up and pull down resistors at the device pin level, according to the Table 2.

Four kinds of resistors are specified and their characteristics are defined in the Table 77.

Table 77. Internal Pin Pull-up/Pull-down Resistor Electrical Specifications

Characteristics noted under conditions -40 °C < T_A < +125 °C, unless otherwise noted. Typical values noted reflect the approximate parameter means at T_A = 25 °C under nominal conditions, unless otherwise noted.

Symbol	Characteristic	Min	Typ	Max	Unit	Notes
R_{W_PU}	Pin weak pull-up resistor	200	480	800	kΩ	
R_{PU}	Pin pull-up resistor	50	120	200	kΩ	
R_{W_PD}	Pin weak pull-down resistor	200	480	800	kΩ	
R_{PD}	Pin pull-down resistor	50	120	200	kΩ	

6.12.1 Startx Pins Pull-up and Pull-down

The pull resistor direction for the STARTx pins can be configurable as pull-up or pull-down.

The configuration of the pull resistor direction is done at device initialization. The direction of the pull resistors is selected according to the polarity of the STARTx signal. This guarantees the pull resistor is used to move the input pin to its inactive state in case of e.g. a broken PCB track.

The pull resistor is configured according to the Flag_polarity register (0x1C2). If the STARTx pins are used as primary function (i.e. start function) os as flag pins the corresponding pull resistor direction is configured as shown in the Table 78

Table 78. STARTx Pin Pull Resistor Direction Selection

STARTx Pin Polarity (star_polx/flag_pol_x)	Pull-down Active	Pull-up active
0	yes	no
1	no	yes

6.13 Unused Pins Connection

Except for supplies and grounds, the application circuits can leave device pins unconnected without any impact on the device for digital I/O signals, OA_1 and OA_2 pins, HS and LS pre-drivers and current measurement channels.

The current measurement channels VSENSEPx and VSENSENx input pins can be connected both to GND.

The table Table 79 details the recommended connections in case of unused pins.

Table 79. Recommended Unused Pins Connections

Pin Number	Pin Name	Recommended Unused Pins Connection
1	CLK	Not connected- internal weak pull-up
2	DRVEN	Not connected - internal weak pull-down
3	RESETB	Not connected - internal weak pull-up
4	START1	Not connected - internal configurable pull-up/pull-down
5	START2	Not connected - internal configurable pull-up/pull-down
6	START3	Not connected - internal configurable pull-up/pull-down
7	START4	Not connected - internal configurable pull-up/pull-down
8	START5	Not connected - internal configurable pull-up/pull-down
9	START6	Not connected - internal configurable pull-up/pull-down

Table 79. Recommended Unused Pins Connections (continued)

Pin Number	Pin Name	Recommended Unused Pins Connection
10	FLAG0	Not connected - internal weak pull-down
11	FLAG1	Not connected - internal weak pull-down
12	FLAG2	Not connected - internal weak pull-down
13	CSB	Not connected - internal pull-up
14	MOSI	Not connected - internal weak pull-up
15	MISO	Not connected
16	SCLK	Not connected - internal weak pull-up
17	VCCIO	To be supplied - filtering capacitor required
18	DBG	Not connected - internal weak pull-up
19	DGND	Connection to ground required
20	VCC2P5	To be supplied - filtering capacitor required
21	VCC5	To be supplied - filtering capacitor required
22	OA_1	Not connected(85) - internal weak pull-down
23	OA_2	Not connected(85) - internal weak pull-down
24	AGND	Connection to ground required
25	VSENSEN1	Not connected(86) (87)
26	VSENSEP1	Not connected(86) (87)
27	VSENSEN2	Not connected(86) (87)
28	VSENSEP2	Not connected(86) (87)
29	VSENSEN3	Not connected(86) (87)
30	VSENSEP3	Not connected(86) (87)
31	VSENSEN4	Not connected(86) (87)
32	VSENSEP4	Not connected(86) (87)
33	D_LS6	Not connected(87)
34	D_LS5	Not connected(87)
35	D_LS4	Not connected(87)
36	D_LS3	Not connected(87)
37	D_LS2	Not connected(87)
38	D_LS1	Not connected(87)
39	VBATT	To be supplied
40	VCCP	Filtering capacitor required
41	G_LS7	Not connected(88)
42	G_LS6	Not connected(88)
43	G_LS5	Not connected(88)
44	G_LS4	Not connected(88)
45	G_LS3	Not connected(88)
46	G_LS2	Not connected(88)
47	G_LS1	Not connected(88)
48	VBOOST	To be supplied
49	B_HS5	Not connected(89)

Table 79. Recommended Unused Pins Connections (continued)

Pin Number	Pin Name	Recommended Unused Pins Connection
50	G_HS5	Not connected(88)(89)
51	S_HS5	Not connected(87)(89)
52	B_HS4	Not connected(89)
53	G_HS4	Not connected(88)(89)
54	S_HS4	Not connected(87)(89)
55	B_HS3	Not connected(89)
56	G_HS3	Not connected(88)(89)
57	S_HS3	Not connected(87)(89)
58	B_HS2	Not connected(89)
59	G_HS2	Not connected(88)(89)
60	S_HS2	Not connected(87)(89)
61	B_HS1	Not connected(89)
62	G_HS1	Not connected(88)(89)
63	S_HS1	Not connected(87)(89)
64	IRQB	Not connected - internal weak pull-down
Exposed pad	PGND	Connection to ground required

Notes

85. Setting the AO_x pin as flag output is recommended.
86. The VSENSEPx and VSENSENx pins can be connected to ground.
87. The crossbar switch must be set up such as to prevent the microcores to be enabled by the function.
88. The crossbar switch must be set up such as to prevent the microcores to enable the function.
89. Not connected if the related driver is not used.

6.14 Internal Digital Signals Description

The main digital signal used into the analog resources blocks are described into the Table 80.

Table 80. Main Internal Digital Signal Description

Signal Name	Bit Bus Size	Description	Signal Block Generator	Signal Block User
Bandgap Reference				
bg_ok	1	This signal is set high when the bandgap voltage is inside its expected range.	Band gap reference	Power On Reset
VCC2P5 and POR				
POResetB	1	This signal is set to high when the VCC2P5 is below its undervoltage lockout threshold.	VCC2P5 and Power On Reset	Power On Reset
SPIResetB	1	This signal issued from the SPI block is set to its low state while the SPI block is in reset.	SPI interface	Power On Reset
ResetB	1	This signal is a living copy of the RESETB pin state.	RESETB pin	Power On Reset block
RSTB	1	This signal is issues from the combination of POResetB, SPIresetB and ResetB (AND combination	Power On Reset	Logic Core

Table 80. Main Internal Digital Signal Description (continued)

Signal Name	Bit Bus Size	Description	Signal Block Generator	Signal Block User
VCC Monitoring				
uv_vcc5	1	This signal is set high if the VCC5 voltage is below its undervoltage lockout threshold.	VCC5 block	High-side and low-side pre-drivers
VCCP Monitoring				
uv_vccp	1	This signal is set high if the VCCP voltage is below its undervoltage lockout threshold	VCC5 block	High-side and low-side pre-drivers
vccp_external_enable	1	This signal disable the VCCP internal regulator	VCCP LDO	Digital block
DC-DC Converter				
boost_fbk	1	This signal it the boost voltage output comparator.	Boost Voltage Monitoring	Logic Core
dac_boost_value	8	This signal is the output of the boost voltage monitoring comparator.	Boost Voltage Monitoring	Boost Voltage Monitoring
vboost_disable_en	1	This signal enables a function that automatically disable all the high-side pre-drivers if the boost voltage is below its undervoltage threshold. This function is activated if the boost monitoring block is set in the Vboost UV Monitoring mode. This bit can be set in the driver_disable register (0x1C5).	Boost Voltage Monitoring	Logic Core
uv_vboost	1	This signal is used to automatically disable all the high-side pre-drivers if the boost voltage is below its undervoltage threshold. This signal is a living copy of boost_fkb and is activated only if the boost monitoring block is set in the Vboost UV Monitoring mode. This bit is reported in the Driver_status register (0x1D2).	Boost Voltage Monitoring	Logic Core
boost_mon_en	1	This signal is used to selected the boost voltage monitoring mode. According to its state the boost divider ratio is typically 1/4 or 1/32. This bit can be set in the Driver_config register (0x1C5).	Logic Core	Boost Voltage Monitoring
Temperature Monitoring				
over_temp	1	This signal is asserted if the internal temperature threshold is exceeded. This bit is reported in the Driver_status register (0x1D2).	Temperature Monitoring	Logic Core
High-side and Low-side Pre-drivers				
hsx_command(90)	1 x 5	These signals are the pre-drive command issued from the combination of the digital driver command (hsx_in), uv_vccp, uv_vcc5 and cksys_drven. These signals directly control the pre-drivers.	High-side Pre-drivers	High-side Pre-drivers
hsx_in(90)	1 x 5	These signals are the pre-driver commands issued from the digital block.	Logic Core	High-side Pre-drivers

Table 80. Main Internal Digital Signal Description (continued)

Signal Name	Bit Bus Size	Description	Signal Block Generator	Signal Block User
lsx_ command(91)	1 x 6	These signals are the pre-drive commands issued from the combination of the digital driver command issued (hsx_in), uv_vccp, uv_vcc5 and cksys_drven. These signals directly control the pre-drivers.	Low-side Pre-drivers	Low-side Pre-drivers
lsx_in(91)	1 x 6	This signal is the pre-driver command issued from the digital block.	Logic Core	Low-side Pre-drivers
ls7_command	1	This signal is the pre-drive command issued from the combination of the command issued from the digital block (hsx_in), uv_vccp, uv_vcc5 and cksys_drven. This signal directly controls the pre-driver.	Low-side pre-driver 7	Low-side pre-driver 7
ls7_in	1	This signal is the pre-driver command issued from the digital block.	Logic Core	Low-side pre-driver 7
ls7_slewrate_ p	2	This signal bus determines the pre-driver seven slew rate. This signal bus can be set by means of the bits slewrate_ls7_rising(1:0) in the Ls_slewrate register (0x18F).	Logic Core	Low-side pre-driver 7
ls7_slewrate_ n	2	This signal bus determines the pre-driver seven slew rate. This signal bus can be set by means of the bits slewrate_ls7_falling(1:0) in the Ls_slewrate register (0x18F).	Logic Core	Low-side pre-driver 7

VDS and VSRC Monitoring

Signal Name	Bit Bus Size	Description	Signal Block Generator	Signal Block User
hsx_bs_inb	1 x 5	This signal is asserted to disable the bootstrap capacitor charging during End Of Drive phase.	Logic Core	Bootstrap diode
hsx_vds_threshold(90)	3 x 5	This bus signal determines the high-side pre-driver VDS voltage monitoring DAC value. The DAC value is set in the Vds_threshold_hs register (0x18A).	Logic Core	High-side Pre-drivers
hsx_src_threshold(90)	3 x 5	This bus signal determines the high-side pre-driver VSRC voltage monitoring DAC value. The DAC value is set in the Vsrc_threshold_hs register (0x18B).	Logic Core	High-side Pre-drivers
lsx_vds_threshold(91)	3 x 6	This bus signal determines the low-side pre-driver VDS voltage monitoring DAC value. The DAC value is set in the Vds_threshold_ls_1 and Vds_threshold_ls_1 registers (0x18C and 0x18D).	Logic Core	Low-side Pre-drivers
hsx_bias(90)	1 x 5	These control signals activate the high-side pre-driver biasing structures for each driver.	Logic Core	High-side Pre-drivers
hsx_bias_strong(92)	1 x 2	These control signals activate the high-side pre-driver strong biasing structures for the HS2 and HS4 pre-drivers.	Logic Core	High-side Pre-drivers
lsx_bias(91)	1 x 6	These control signals activate the low-side pre-driver biasing structures for each driver.	Logic Core	Low-side Pre-drivers

Current Measurement Blocks and OA_x Outputs

Signal Name	Bit Bus Size	Description	Signal Block Generator	Signal Block User
opamapx_ gain(93)	2 x 4	These signal bus determine the operational amplifier gain value for each of the current sense differential amplifiers.	Logic Core	Current Measurement Blocks

Table 80. Main Internal Digital Signal Description (continued)

Signal Name	Bit Bus Size	Description	Signal Block Generator	Signal Block User
oa_gainx (or oa_gainy)(94)	2	This signal bus determines the OA_1 and OA_2 buffer gain.	Logic Core	Current Measurement Blocks
dacx_value	5 x 8	This signal bus provides the DAC values to each current measurement block DACs. The DAC values are set in the Dacx_value registers (0x19E, 0x19F, 0x1A0, 0x1A1, 0x1A2).	Logic Core	Current Measurement Blocks
dac4neg_ value	3	This signal bus is the DAC value provided to the negative comparator of the current measurement block 4. The DAC value is set in the Dac4neg_value register (0x1A)).	Logic Core	Current Measurement Blocks
curx_fbk(95)	6 x 1	These signals are the current measurement block comparator outputs	Current Measurement Blocks	Logic Core
cur4h_fbk	1	This signal is the current measurement block 4 high current comparator output.	Current Measurement Block 4	Logic Core
cur4l_fbk	1	This signal is the current measurement block 4 low current comparator output.	Current Measurement Block 4	Logic Core
OaSel1	3	This signal bus controls the OA_1 pin output multiplexer.	Logic Core	OA_x Output
OaSel2	3	This signal bus controls the OA_2 pin output multiplexer.	Logic Core	OA_x Output
SPI				
miso_ slewrate	1	This bit is set into the SPI_config register (0x1C8) such as to configure the SPI in slow or fast mode	Devices internal registers	SPI interface
PLL and Clock Monitoring				
cksys_drven	1	This signal is asserted low when a missing external clock condition is detected. This signal is asserted low until the device has switched to its internal backup clock.	Logic Core	High-side and Low-side Pre-drivers
cksys_missing	1	This signal is set when the internal PLL doesn't provide a valid clock.	PLL and Backup Clock	Logic Core
cksys_drven	1	This signal enables the function that automatically disable all the high-side pre-drivers while the cksys_missing signal is asserted.	PLL and Backup Clock	High-side and Low-side Pre-drivers
Drive Enable				
DrvEn	1	This signal is a live copy of the DRVEN pin state.	DRVEN pin	High-side and Low-side Pre-drivers

Notes

90. These signals or signal bus are individually provided to the five high-side pre-drivers (x= 1, 2, 3, 4, or 5).
91. These signals or signal bus are individually provided to the first six high-side pre-drivers. Low-side pre-driver seven is considered independently. (x= 1, 2, 3, 4, 5, or 6).
92. The strong biasings are only available on the high-side pre-drivers 2 and 4.
93. These signals or signal bus are individually provided to the four current measurement blocks (x= 1, 2, 3, or 4).
94. (x or y = 1 or 2)
95. These signals re individually produced by the four current measurement blocks (x= 1, 2, 3, or 4).

6.15 Device Logic Block Description

The 33816 provides a logic block called two_ channels, manages the analog resources and interfaces with the MCU. This block is the hierarchical top level of the digital part of the device.

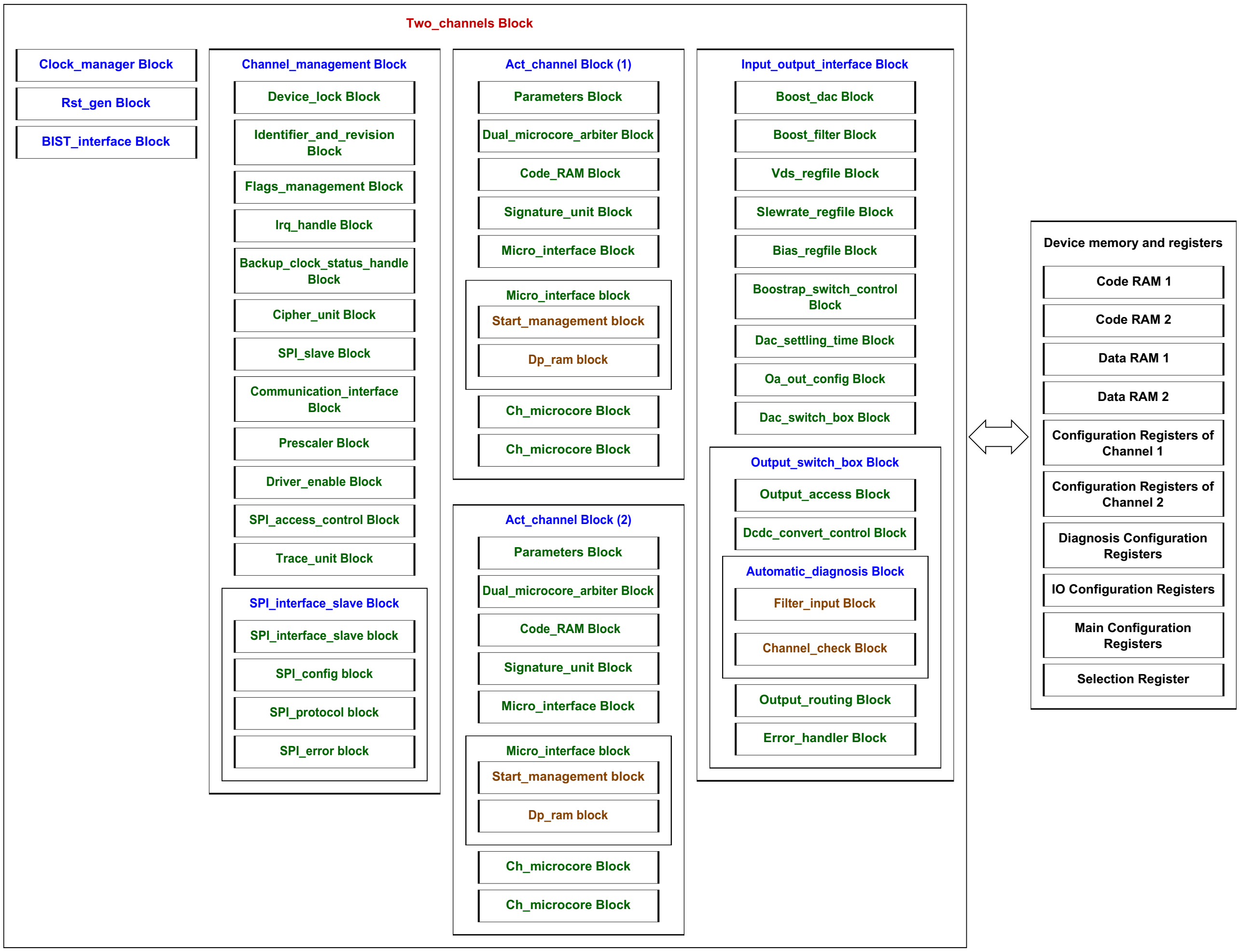


Figure 24. Device Logic Block Hierarchic Overview

The two_channels block manages the cores functions, such as clock signals, reset, and built-in self-test. The two_channels block integrates the following sub-blocks:

- clock_manager – this block is dedicated to the device clock management
- rst_gen _ this block is in charge of device reset management
- bist_interface – this block is in charge of managing the BIST function
- act_channel – this block is in charge to handle two microcore and the associated functions like Code RAM access and dual sequencing.
- channel_management – this block is dedicated to the act_channel blocks signal management, managing the common configuration setup and managing the communication interfaces of the 33816
- input_output_interface (refer to IO interface section)

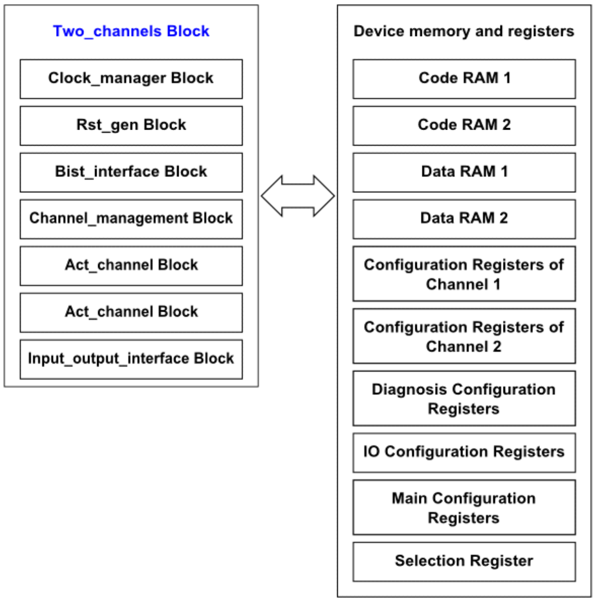


Figure 25. Two_channels Block Overview

The logic block setup is managed through banks of registers that can be access through the SPI. Each bank is dedicated to a specific functionality of the logic block a shown below:

- Configuration Registers of Channel 1 and Channel 2- These register areas are dedicated to the logic channels 1 and 2 setup.
- Diagnosis Configuration Registers – These registers are used to setup the automatic diagnosis parameters and the diagnostic option.
- IO Configuration Registers – These registers are basically used to setup the crossbar switch and all the current and voltage thresholds used by the analog resources.
- Main Configuration registers – these registers are dedicated to the logic block functionality setup, like the clock management, the flags properties, SPI management, the trace management, and the other peripheral functions.

All the registers and their corresponding addresses are listed in the Device Address Map table.

6.15.1 33816 Address Map

Table 81. Device Address Map

Selection register [2:0]/Chip select	Address (Dec)	Address (Hex)	Lock [(96)]	Description/Name	Area Addressed
“001” /ch_sel_1(1)	0	0	yes	Code RAM of channel 1	Code RAM of Channel 1
	…	…			
	1022	0x3FE			
“010”/ ch_sel_2(1)	0	0	yes	Code RAM of channel 2	Code RAM of Channel 2
	…	…			
	1022	0x3FE			

Table 81. Device Address Map (continued)

Selection register [2:0]/Chip select	Address (Dec)	Address (Hex)	Lock [(96)]	Description/Name	Area Addressed
“100”/ch_sel_1(0)	0	0	no	Data RAM of channel 1	Data RAM of Channel 1
	…	…			
	47	0x02F			
	48	0x030	yes	Data RAM of channel 1, private area	
	...	…			
	63	0x03F			
“100”/ch_sel_2(0)	64	0x040	no	Data RAM of channel 2	Data RAM of Channel 2
	…	…			
	111	0x06F			
	112	0x070	yes	Data RAM of channel 2, private area	
	…	…			
	127	0x07F			
“100”/ch_sel_1(2)	128	0x080		128 reserved addresses	Configuration Registers of Channel 1
	…	…			
	255	0x0FF			
	256	0x100	yes	Flash_enable of channel 1	
	257	0x101	no	Ctrl_reg_uc0 of channel 1	
	258	0x102	no	Ctrl_reg_uc1 of channel 1	
	259	0x103	no	Unlock_word of channel 1	
	260	0x104	yes	Start_config_reg of channel 1	
	261	0x105	-	Status_reg_uc0 of channel 1	
	262	0x106	-	Status_reg_uc1 of channel 1	
	263	0x107	yes	Code_width of channel 1	
	264	0x108	yes	Checksum_h of channel 1	
	265	0x109	yes	Checksum_l of channel 1	
	266	0x10A	yes	Uc0_entry_point of channel 1	
	267	0x10B	yes	Uc1_entry_point of channel 1	
	268	0x10C	yes	Diag_routine_addr of channel 1	
	269	0x10D	yes	Driver_disabled_routine_addr of channel 1	
	270	0x10E	yes	Sw_interrupt_routine_addr of channel 1	
	271	0x10F	no	Uc0_irq_status of channel 1	
	272	0x110	no	Uc1_irq_status of channel 1	
	273	0x111	yes	Counter_34_prescaler of channel 1	
	274	0x112	yes	Control_register_split of channel 1	
	275	0x113		13 free addresses	
	…	…			
	287	0x11F			

Table 81. Device Address Map (continued)

Selection register [2:0]/Chip select	Address (Dec)	Address (Hex)	Lock [96]	Description/Name	Area Addressed
“100”/ch_sel_2(2)	288	0x120	yes	Flash_enable of channel 2	Configuration Registers of Channel 2
	289	0x121	no	Ctrl_reg_uc0 of channel 2	
	290	0x122	no	Ctrl_reg_uc1 of channel 2	
	291	0x123	no	Unlock_word of channel 2	
	292	0x124	yes	Start_config_reg of channel 2	
	293	0x125	-	Status_reg_uc0 of channel 2	
	294	0x126	-	Status_reg_uc1 of channel 2	
	295	0x127	yes	Code_width of channel 2	
	296	0x128	yes	Checksum_h of channel 2	
	297	0x129	yes	Checksum_l of channel 2	
	298	0x12A	yes	Uc0_entry_point of channel 2	
	299	0x12B	yes	Uc1_entry_point of channel 2	
	300	0x12C	yes	Diag_routine_addr of channel 2	
	301	0x12D	yes	Driver_disabled_routine_addr of channel 2	
	302	0x12E	yes	Sw_interrupt_routine_addr of channel 2	
	303	0x12F	no	Uc0_irq_status of channel 2	
	304	0x130	no	Uc1_irq_status of channel 2	
	305	0x131	yes	Counter_34_prescaler of channel 2	
	306	0x132	yes	Control_register_split of channel 2	
	307	0x133		13 reserved addresses	
	…	…			
	319	0x13F			
“100”/ext_sel_diag	320	0x140	yes	Ls1_diag_config1	Diagnosis Configuration Registers
	321	0x141	yes	Ls1_diag_config2	
	322	0x142	yes	Ls1_output_config	
	323	0x143	yes	Ls2_diag_config1	
	324	0x144	yes	Ls2_diag_config2	
	325	0x145	yes	Ls2_output_config	
	326	0x146	yes	Ls3_diag_config1	
	327	0x147	yes	Ls3_diag_config2	
	328	0x148	yes	Ls3_output_config	
	329	0x149	yes	Ls4_diag_config1	
	330	0x14A	yes	Ls4_diag_config2	
	331	0x14B	yes	Ls4_output_config	

Table 81. Device Address Map (continued)

Selection register [2:0]/Chip select	Address (Dec)	Address (Hex)	Lock [(96)]	Description/Name	Area Addressed
“100”/ext_sel_diag	332	0x14C	yes	Ls5_diag_config1	Diagnosis Configuration Registers
	333	0x14D	yes	Ls5_diag_config2	
	334	0x14E	yes	Ls5_output_config	
	335	0x14F	yes	Ls6_diag_config1	
	336	0x150	yes	Ls6_diag_config2	
	337	0x151	yes	Ls6_output_config	
	338	0x152	yes	Ls7_output_config	
	339	0x153	yes	Hs1_diag_config_1	
	340	0x154	yes	Hs1_diag_config_2	
	341	0x155	yes	Hs1_output_config	
	342	0x156	yes	Hs2_diag_config_1	
	343	0x157	yes	Hs2_diag_config_2	
	344	0x158	yes	Hs2_output_config	
	345	0x159	yes	Hs3_diag_config_1	
	346	0x15A	yes	Hs3_diag_config_2	
	347	0x15B	yes	Hs3_output_config	
	348	0x15C	yes	Hs4_diag_config_1	
	349	0x15D	yes	Hs4_diag_config_2	
	350	0x15E	yes	Hs4_output_config	
	351	0x15F	yes	Hs5_diag_config_1	
	352	0x160	yes	Hs5_diag_config_2	
	353	0x161	yes	Hs5_output_config	
	354	0x162	-	Err_uc0ch1_1	
	355	0x163	-	Err_uc0ch1_2	
	356	0x164	-	Err_uc1ch1_1	
	357	0x165	-	Err_uc1ch1_2	
	358	0x166	-	Err_uc0ch2_1	
	359	0x167	-	Err_uc0ch2_2	
	360	0x168	-	Err_uc1ch2_1	
	361	0x169	-	Err_uc1ch2_2	
	362	0x16A	yes	Fw_ext_req	
	363	0x16B	yes	Diagnosis_option	
	364	0x16C		20 reserved addresses	
	…	–			
	383	0x17F			

Table 81. Device Address Map (continued)

Selection register [2:0]/Chip select	Address (Dec)	Address (Hex)	Lock [96]	Description/Name	Area Addressed
“100”/ext_sel_io	384	0x180	yes	Fbk_sens_uc0_ch1	IO Configuration Registers
	385	0x181	yes	Fbk_sens_uc1_ch1	
	386	0x182	yes	Fbk_sens_uc0_ch2	
	387	0x183	yes	Fbk_sens_uc1_ch2	
	388	0x184	yes	Out_acc_uc0_ch1	
	389	0x185	yes	Out_acc_uc1_ch1	
	390	0x186	yes	Out_acc_uc0_ch2	
	391	0x187	yes	Out_acc_uc1_ch2	
	392	0x188	yes	Cur_block_access_1	
	393	0x189	yes	Cur_block_access_2	
	394	0x18A	no	Vds_threshold_hs	
	395	0x18B	no	Vsrc_threshold_hs	
	396	0x18C	no	Vds_threshold_ls_1	
	397	0x18D	no	Vds_threshold_ls_2	
	398	0x18E	no	Hs_slewrate	
	399	0x18F	no	Ls_slewrate	
	400	0x190	no	Offset_compensation1	
	401	0x191	no	Offset_compensation2	
	402	0x192	no	Offset_compensation3	
	403	0x193	no	Offset_compensation4	
	404	0x194	no	Adc1_result	
	405	0x195	no	Adc2_result	
	406	0x196	no	Adc3_result	
	407	0x197	no	Adc4_result	
	408	0x198	yes	Current_filter12	
	409	0x199	yes	Current_filter34l	
	410	0x19A	yes	Current_filter4h4neg	
	411	0x19B	no	Boost_dac	
	412	0x19C	yes	Boost_dac_access	
	413	0x19D	yes	Boost_filter	
	414	0x19E	no	Dac1_value	
	415	0x19F	no	Dac2_value	
	416	0x1A0	no	Dac3_value	
	417	0x1A1	no	Dac4l_value	
	418	0x1A2	no	Dac4h_value	
	419	0x1A3	no	Dac4neg_value	

Table 81. Device Address Map (continued)

Selection register [2:0]/Chip select	Address (Dec)	Address (Hex)	Lock [(96)]	Description/Name	Area Addressed
“100”/ext_sel_io	420	0x1A4	no	Bias_config	IO Configuration Registers
	421	0x1A5	-	Bootstrap_charged	
	422	0x1A6	yes	Hs12_ls_act	
	423	0x1A7	yes	Hs34_ls_act	
	424	0x1A8	yes	Hs5_ls_act	
	425	0x1A9	yes	Dac_settling_time	
	426	0x1AA	no	Oa_out1_config	
	427	0x1AB	no	Oa_out2_config	
	428	0x1AC		20 reserved addresses	
	…	–			
	447	0x1BF			
“100”/ext_sel_mcr	448	0x1C0	yes	Ck_per	Main Configuration Registers
	449	0x1C1	yes	Flags_direction	
	450	0x1C2	yes	Flags_polarity	
	451	0x1C3	yes	Flags_source	
	452	0x1C4	yes	Ck_ofscomp_per	
	453	0x1C5	yes	Driver_config	
	454	0x1C6	yes	PLL_config	
	455	0x1C7	yes	Backup_clock_status_reg	
	456	0x1C8	yes	SPI_config	
	457	0x1C9	yes	Reserved	
	458	0x1CA	no	Trace_start	
	459	0x1CB	no	Trace_stop	
	460	0x1CC	no	Trace_config	
	461	0x1CD	yes	Device_lock	
	462	0x1CE	no	Reset_behavior	
	463	0x1CF	-	Device_unlock	
	464	0x1D0	-	Global_reset, part 1	
	465	0x1D1	-	Global_reset, part 2	
	466	0x1D2	no	Driver_status	
	467	0x1D3	-	SPI_error	
	468	0x1D4	no	Interrupt_status	
	469	0x1D5	-	Identifier_revision	
	470	0x1D6	-	Reset_source	
	471	0x1D7		Reserved	
	472	0x1D8		Reserved	
	473	0x1D9		Reserved	

Table 81. Device Address Map (continued)

Selection register [2:0]/Chip select	Address (Dec)	Address (Hex)	Lock [96]	Description/Name	Area Addressed
“100”/ext_sel_mcr	474	0x1DA		Reserved	Main Configuration Registers
	475	0x1DB		Reserved	
	476	0x1DC	no	BIST_interface	
	477	0x1DD		Reserved	
	478	0x1DE		Reserved	
	479	0x1DF		33 reserved addresses	
	…	–			
	511	0x1FF			
	512	0x200		511 reserved addresses	
	…	–			
	1022	0x3FE			
	1023	0x3FF	no	Selection register (Selection_reg register)	

Notes

96. The memory areas and register can be locked according to the table column Lock = Yes by means of the Device_lock register (0x1CD).

6.15.2 Clock_manager Block

To generate the system clock cksys, the device has a PLL with a frequency multiplication factor selectable between the two values 12 and 24. The PLL can be supplied either with an external reference in the f_{CLK} frequency range or with an internal backup reference in the f_{CLK_BACK} frequency range.

The cksys feeding all the internal logic is disabled while the filtered pll_lock signal is set to ‘0’. The filter length for the pll_lock signal is $t_{PLL\_LOCK\_FILTER}$. This clock_manager block is in charge of:

- selecting between the two input references for the PLL.
- providing a loss_of_clock signal to channel_management.This signal is ‘1’ when the backup reference is supplied to the PLL, a ‘0’ otherwise. This status can be read back through the SPI in the Backup_clock_status_reg register (0x1C7).
- providing a cksys_missing signal to channel_management. This signal is ‘1’ when the PLL provides no valid clock signal, a ‘0’ otherwise. This signal is used to generate an interrupt request to the microcores and to the external microcontroller through the IRQB pin. The signal cksys_missing can also be used to disable the output drivers if the bit cksys_missing_disable_driver bit is set to ‘1’ in Backup_clock_status_reg register (0x1C7).

Considering the MCU might not be able to provide a stabilized clock to the CLK pin of the device during its reset the clock manager, FSM is disabled for 100 μs. After that, the RESETB pin signal is set to low. While in this state, the external reference is provided to the PLL input and no check is performed on the PLL output. The loss_of_clock and cksys_missing signals are deactivated. After filter time is reached, the clock monitor is enabled and can detect a low frequency or missing clock input reference, or a PLL malfunction.

This sub-block monitors the PLL output to detect any output clock frequency out of the expected range. This is achieved by counting the number of pll_output_clock cycles inside six periods of the backup reference clock (6 period of 1.0 MHz clock = 6.0 μs). The expected number depends on the selected PLL multiplication factor:

- when the factor is 24, it detects an invalid clock condition when it is possible to count more than 165 or less than 125 pll_output_clock cycles.
- when the factor is 12, it detects an invalid clock condition when it is possible to count more than 84 or less than 61 pll_output_clock cycles.

If an invalid clock condition is detected, the fsm clock_manager goes to the “Locking onto Backup Reference” state. In this state, the backup reference is supplied to the PLL and both loss_of_clock and cksys_missing signals are set. When the PLL locks onto the backup reference, the fsm clock_manager goes to the “Use Backup Reference” state. In this state, the loss_of_clock signal is still set while the cksys_missing signal is reset.

While the fsm clock_manager is in “Use Backup Reference” state, it is possible to recheck the external clock reference. The check is not performed automatically, but must be requested by writing to the Backup_clock_status_reg register (0x1C7).

When the switch back is requested (“Switch to clock pin”), the cksys_missing is asserted, the loss of clock is reset, and the clock monitor is disabled again for 250 µs. The cksys_missing signal is active at least until the PLL is locked onto the external clock reference. The PLL is considered locked again if one of two conditions are verified:

- A falling edge followed by a rising edge on the pll_out_valid signal, which includes the information from the frequency counter and the filtered pll_lock signal.
- A fixed time of 100 µs has passed since the switch has been made and the pll_out_valid signal was always active.

When one of the conditions is verified, the cksys_missing signal is reset and the fsm goes to “Use External reference” state. After requesting the switch back to the external clock reference, the device cannot be accessed via the SPI for about:

- 100 µs, if there is a valid external clock available
- 290 µs, if there is no valid input clock available and the device has to return to the backup clock.

The SPI word transmitted to set the switch to clock pin bit has to be the last word within a SPI burst.

6.15.2.1 PLL Configuration Register

Table 82. PLL_config Register (0x1C6)

Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Reserved														PLL_ spread_ disable	PLL_ factor
R/W	-														r/w	r/w
Lock	-														yes	yes
Reset	0000000000000														0	1

- PLL_factor: if set to ‘0’, the PLL multiplication factor is 12, otherwise it is 24
- PLL_spread_disable: if set to ‘0’ spread is applied to the PLL output clock, otherwise spread is disabled

The PLL factor is changed synchronously with clock monitor cycle to avoid a clock monitor alert when changing between 12 and 24 MHz.

6.15.3 Rst_gen Block

The device has three sources of reset:

- low signal on RESETB pin
- the internal signal POResetB, generated by the V_{CC2P5} voltage regulator undervoltage flag
- a global reset request received through the SPI, writing the reset code into the Global_reset Registers (0x1D0 and 0x1D1)

This block generates the following reset signals:

- Clock Monitor reset. This signal is activated asynchronously when either ResetB or POResetB are activated. It is deactivated synchronously with the backup clock reference when both ResetB and POResetB are inactive. This reset is supplied to the clock_manager block. This reset differs from the others, as it is synchronized to the backup clock reference.
- Cipher Register reset. This signal is activated asynchronously when the POResetB is activated. It is deactivated synchronously with the cksys clock when the POResetB is inactive. This reset is supplied to the cipher configuration register. This reset puts the cipher configuration register in the same condition of the RAMs, which are reset only in case of a power loss.
- SPI interface reset. This signal is activated asynchronously when either the ResetB, POResetB, global SPI reset, or cksys_missing is activated. This reset signal is deactivated synchronously with the cksys clock when all the signals ResetB, POResetB, global SPI reset, cksys_missing are inactive and the SPI chip select is inactive. If any SPI transfer is required while thecksys_missing signal is active, then the SPI transfer is aborted and a SPI error is stored in the SPI_error register (0x1D3).This condition avoids a transfer of dummy data, leading to a hieratic device operation.

- Main reset (rst). This signal is activated asynchronously when the ResetB, POResetB, or global SPI reset is activated. It is deactivated synchronously with the cksys clock when ResetB, POResetB, and global SPI reset are inactive. This signal resets all the device, except the blocks using other resets.

6.15.3.1 Global Reset Registers

This 32-bit register is divided into two 16-bit slices. When the correct "global reset code" is written in this register, the rst_gen block forces the entire device in reset, except for the rst_gen block itself, the clock manager block, and the cipher configuration register. This reset lasts for eight cksys clock cycles, then the global reset registers are reset.

The global reset code is '0xF473' for Global reset register 1 and '0x57A1' for Global reset register 2.

Table 83. Global_reset Register 1 (0x1D0)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	global_reset_code_1															
R/W	w															
Lock	no															
Reset	0x0000															

Table 84. Global_reset Register 2 (0x1D1)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	global_reset_code_2															
R/W	w															
Lock	no															
Reset	0x0000															

6.15.3.2 Reset Source Register

This 3-bit register identifies which resets were asserted since the last time this register was read. This register is reset at each access.

Table 85. Reset_source Register (0x01D6)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved													SPI_ reset	POResetB	resetB
R/W	-													r	r	r
Lock	-													-	-	-
Reset on read	-													yes	yes	yes
Reset	0000000000000													*	*	*

SPI _reset is '1' if the global SPI reset was asserted since the last time this register was read.

POReseB is '1' if the power on reset was asserted since the last time this register was read.

ResetB is '1' if the reset pin was asserted since the last time this register was read. After POResetB this bit is in an unknown state.

6.15.4 BIST_interface Block

A full BIST check of the device memories can be enabled accessing the BIST_register in write mode and writing a 16-bit password ‘0xB157’ (BIST_activation_password). This request is accepted only if both Code RAM 1 and Code RAM 2 are unlocked, writing the code ‘0xBEEF into the Unlock_word register of the channel 1 (0x103) and Unlock_word register of the channel 2 (0x123). After this request is performed, the BIST check starts and can be monitored at any time, accessing the same BIST_register in read mode.

The overall BIST operation takes about 2.2 ms (at 24 MHz) to complete. During the memory BIST, five different tests are performed using different patterns to test the RAM. The patterns are:

- All 00, All 11
- All 55, All AA
- All 0F, All F0
- All 00, All FF
- All FF, All 00

Table 86. BIST_interface Register in Write Mode (0x1DC)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BIST_activation_password															
R/W	w															
Lock	no															
Reset	-															

Table 87. BIST_interface in Read Mode (0x1DC)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved														BIST_result	
R/W	-														r	
Lock	-														no	
Reset	00000000000000														00	

- BIST_result: set to “00” if the BIST has never been requested
- BIST_result: set to “01” if the BIST operation is in progress
- BIST_result: set to “10” if the BIST operation has been successfully completed
- BIST_result: set to “11” if the BIST operation has failed

6.15.5 Channels_Management Block

This General Channels Management block called channel_management provides the following services:

- combines all the signals issued from or feeding to the 2 act_channel blocks
- provides all the main configuration registers
- interfaces the 33816 with the MCU through the SPI connection.

The channel_management block integrates the following blocks:

- Device_lock
- Identifier_and_revision
- Flags_management
- Irq_handle
- Cipher_unit
- SPI_slave
- Communication_interface
- Prescalers
- Driver_enable

- SPI_access_controller
- Trace_unit

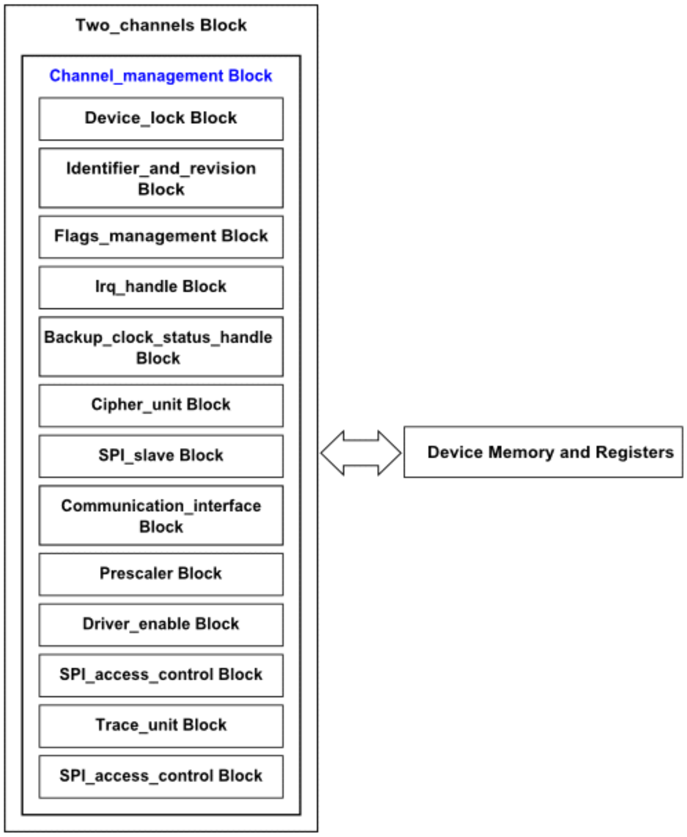


Figure 26. Channel_management Block Diagram

6.15.5.1 Device_lock Block

Some device registers can be protected against an unexpected write. In lock mode, these registers can only be accessed in read mode. The register lock is not mandatory for the device normal operation. This lock mode is mainly dedicated to safety and can be reset at any time while writing an unlock password ‘0x1337’ into the Device_unlock register (0x1CF).

Note that the last 16 addresses of each Data RAM can be independently locked.

6.15.5.1.1 Device_lock Register

Table 88. Device_lock Register (0x1CD)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved													dram2_private_area_lock	dram1_private_area_lock	dev_lock
R/W	-													r/w	r/w	r/w
Lock	no													yes by itself	yes by itself	yes
Reset	0000000000000													0	0	0

The device lock mode can be enabled by writing ‘1’ in the dev_ lock bit of the Device_lock register. Device lock mode cannot be reset by writing to the device lock register but only by writing the unlock password ‘0x1337’ into the Device_unlock register (0x1CF).

If the device lock bit Dev Lock is set to ‘1’, all of the register that can be locked (see Device Address Map - lock status is 'yes') cannot be changed further by the SPI. Writing the dev_ lock bit as no effect on the last 16 addresses lock of each Data RAM. This two RAM section can only be locked by writing ‘1’ in the dram1_ private_ area_lock and dram2_ private_ area_lock that locks the Data RAM private area 1 and 2 respectively.

6.15.5.1.2 Reset Behavior Register

Some registers of the device can be configured to be reset when read by an external device through the SPI. Read accesses by microcores using the SPI backdoor does not reset those registers.

Table 89. Reset_behavior Register (0x1CE)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved							sr_uc1_ch2_rb	sr_uc0_ch2_rb	sr_uc1_ch1_rb	sr_uc0_ch1_rb	diag_uc1_ch2_rb	diag_uc0_ch2_rb	diag_uc1_ch1_rb	diag_uc0_ch1_rb	driver_enable_rb
R/W	-							r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	-							no	no	no	no	no	no	no	no	no
Reset	000000							0	0	0	0	0	0	0	0	0

- Driver enable reset behavior (driver_enable_rb): if set to ‘1’ Driver_status register is reset on read.
- Automatic diagnosis uc0 ch1 reset behavior (diag_uc0_ch1_rb): if set to ‘1’ diagnosis error registers (0x162 and 0x163) of microcore 0 of channel 1 is reset on read.
- Automatic diagnosis uc1 ch1 reset behavior (diag_uc1_ch1_rb): if set to ‘1’ diagnosis error registers (0x164 and 0x165) of microcore 1 of channel 1 is reset on read.
- Automatic diagnosis uc0 ch2 reset behavior (diag_uc0_ch2 _rb): if set to ‘1’ diagnosis error registers (0x166 and 0x167) of microcore 0 of channel 2 is reset on read.
- Automatic diagnosis uc1 ch2 reset behavior (diag_uc1_ch2_rb): if set to ‘1’ diagnosis error registers (0x168 and 0x169) of microcore 1 of channel 2 is reset on read.
- Status register uc0 ch1 reset behavior (sr_uc0_ch1_rb): if set to ‘1’ the Status register of microcore 0 of channel 1 is reset on read.
- Status register uc1 ch1 reset behavior (sr_uc1_ch1_rb): if set to ‘1’ the Status register of microcore 1 of channel 1 is reset on read.
- Status register uc0 ch2 reset behavior (sr_uc0_ch2_rb): if set to ‘1’ the Status register of microcore 0 of channel 2 is reset on read.
- Status register uc1 ch2 reset behavior (sr_uc1_ch2_rb): if set to ‘1’ the status register of microcore 1 of channel 2 is reset on read.

During a simultaneous register read SPI command and a register write access by any microcore, the refreshed bits status is available at the next external SPI and at the next register read SPI command.

6.15.5.1.3 Device_unlock Register

Table 90. Device_unlock Register (0x1CF)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	unlock_password															
R/W	w															
Lock	-															
Reset	-															

Writing the password ‘0x1337’ in the unlock_password field resets the full Device_lock register (0x1CD).

6.15.5.2 Identifier_and_revision Block

Table 91. Identification_revision Register (0x1D5)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	device_id								mask_id				sw_id			
R/W	r								r				r			
Lock	-								-				-			
Reset	xxxxxxxx								Per factory programming				Per factory programming			

This register provides a device identifier for the component. The three fields are:

- device_id is a constant that identifies the 33816. Its value is ‘0x9D’
- mask_id is a version number of the mask set used for the device
- sw_id is a version number related to the mask set. The value stored in the sw_id field determines the compatibility with the assembler instruction set.

6.15.5.3 Flags_management Block

This block combines the channel_flags_x coming from the 2 act_channel blocks. The two busses are ANDed together to generate the int_flags bus. This signal is handled in the two_channels block to generate the 16-bit flag_bus bus. This block also contains the configuration registers for the flags polarity selection (Flags_polarity register), flag source (Flags_source register), and flags input or output mode selection (Flags_direction register).

The flag_bus is a general purpose 16-bit signal. Each of the two channels drive its output copy (channel_flags_x) of this bus. The two busses are then combined to generate a single int_flags bus.

13 pins of the device can be routed to the flag bus. If the pin is configured (default configuration) for its initial function, the corresponding element of the flag_bus is taken from the int_flags bus. If the pin is configured to be used as a flag, it loses its initial function and it is connected to a fixed element of the flag bus.

In this case, each of the 13 pins can be configured as an input or output pin:

- if configured as an input, the pin signal overwrites the corresponding element of the flag_bus. In this configuration, an anti-glitch filter is applied to the flag. The filter time is three cksys clock cycles (125 ns at 24 MHz)
- If configured as an output, the corresponding element of the flag_bus is taken from the int_flags bus. The flag_bus element is then directly connected to the external pin

The three remaining internal flags int_flags are simply inputs to the two channels. In view of this feature, the channels can exchange data between them through these three signals, while the other 13 flags can be used either as internal flags or as general purpose I/Os that can be read and written by all the channels.

Table 92 defines the I/Os of the device used as part of the flag_bus. The position of the external pin in the flag_bus cannot be reassigned

Table 92. Flag Pin Assignment

Flag number	Pin assigned
0	FLAG0
1	FLAG1
2	FLAG2
3	START1
4	START2
5	START3
6	START4

Table 92. Flag Pin Assignment (continued)

Flag number	Pin assigned
7	START5
8	START6
9	IRQ
10	OA_1
11	OA_1
12	DBG

The channel_flags_x busses are outputs of the two act_channel blocks, while the flags bus is the input to all the two act_channel blocks.

6.15.5.3.1 Flags_source

Out of the 16 flags, 10 have a configurable source. The three MSB flags can only be used to exchange data internally between the microcores. The three LSB flags can only be used as device I/Os.

This 10-bit register allows to configure the source of the corresponding flag, as shown in Table 93.

Table 93. Flags_source Register (0x1C3)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved			flag_ src 12	flag_ src11	flag_ src10	flag_ src9	flag_ src8	flag_ src7	flag_ src6	flag_ src5	flag_ src4	flag_ src3	Reserved		
Related pin				DBG	OA_2	OA_1	IRQB	STAR T6	STAR T5	STAR T4	STAR T3	STAR T2	STAR T1			
R/W	-			r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	-		
Lock	no			yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes		
Reset	000			1	0	0	1	0	0	0	0	0	0	111		

6.15.5.3.2 Flags_direction

This is a 13-bit register where each bit sets the direction of the corresponding flag, as shown in Table 94. This register value is used only for the flags that drive or can be driven by a device pin as specified in the Flags_source register.

Table 94. Flags_direction Register (0x1C1)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved			flag_ dir12	flag_ dir11	flag_ dir10	flag_ dir9	flag_ dir8	flag_ dir7	flag_ dir6	flag_ dir5	flag_ dir4	flag_ dir3	flag_ dir2	flag_ dir1	flag_ dir0
Related pin				DBG	OA_2	OA_1	IRQB	STAR T6	STAR T5	STAR T4	STAR T3	STAR T2	STAR T1	FLAG2	FLAG1	FLAG0
R/W	-			r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	-			yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Reset	000			1	1	1	1	1	1	1	1	1	1	1	1	1

Table 95. flags_source and flags_direction Registers

flags_source(x)	flags_direction(x)	flag_bus(x) Source
0	0/1	The corresponding pin is used for its non-flag function (start, irq, analog OAx, etc). Flag_bus(x) is driven by int_flags(x).
1	0	The corresponding pin is used as an output flag. The device pin is driven by int_flags(x). Flag_bus(x) is driven by int_flags(x).
1	1	The corresponding pin is used as an input flag. The Flag_bus(x) is driven by the device pin.

6.15.5.3.3 Flags_polarity

This is a 13-bit register where each bit sets the polarity of the corresponding flag, as shown in Table 96. The corresponding flag is inverted if a 1 is set. The value of this register is used only for the flags which are driven by or drive a device pin as specified in the flags_source register.

Table 96. Flags_polarity Register (0x1C2)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved			flag_ pol_12	flag_ pol_11	flag_ pol_10	flag_ pol_9	flag_ pol_8	flag_ pol_7	flag_ pol_6	flag_ pol_5	flag_ pol_4	flag_ pol_3	flag_ pol_2	flag_ pol_1	flag_ pol_0
Related pin				DBG	OA_2	OA_1	IRQB	START6	START5	START4	START3	START2	START1	FLAG2	FLAG1	FLAG0
R/W	-			r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	-			yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Reset	000			0	0	0	0	0	0	0	0	0	0	0	0	0

Table 97. Flags_polarity

flags_polarity(x)	flags_bus(x) condition
0	direct
1	inverted

Some bits of this register are used to set the polarity of the start pins when they are not used as flag I/O.

Table 98. Flag_polarity Register for STARTx Pins Not Used as I/Os (0x1C2)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved			-	-	-	-	start_ pol6	start_ pol5	start_ pol4	start_ pol3	start_ pol2	start_ pol1	-	-	-
R/W	-			-	-	-	-	r/w	r/w	r/w	r/w	r/w	r/w	-	-	-
Lock	-			-	-	-	-	yes	yes	yes	yes	yes	yes	-	-	-
Reset	000			0	0	0	0	0	0	0	0	0	0	0	0	0

Table 99. Start_polarity

flags_polarity(x)	flags_bus(x) condition
0	Start active high
1	Start active Low

6.15.5.4 Irq_handle Block

This block combines all requests to issue an interrupt request on the external IRQB pin.

When one of the possible irq sources request an interrupt, the IRQB pin is asserted (driven low). Meanwhile, the Interrupt_status register (0x1D4) latches the status of all the irq sources. If the IRQB pin is already asserted, further interrupt requests do not change the value of the Interrupt_status register (0x1D4). By reading this register through the SPI, it is possible to ascertain the cause of the interrupt request. When none of the possible sources is requesting an interrupt, the IRQB pin is de-asserted and the Interrupt _reg register is cleared.

The possible sources are:

- one of the four microcores of the microcode requests an interrupt. The interrupt source can be configured through the Sw_interrupt_routine_addr registers (0x10E and 0x12E). The related status is reported in the Uc0_irq_status registers (0x10F and 0x12F) and in the Uc1_irq_status registers (0x110 and 0x130)
- the driver enable block disabling the output drivers. The interrupt source can be configured through the Driver_config register (0x1C5). The related status is reported in the Driver_status register (0x1D2)
- an error that occurs on the SPI interface. The interrupt source can be configured through the SPI_config register (0x1C8). The related status is reported in the SPI_error register (0x1D3)
- the loss of the external clock. For more details, refer to the Clock_manager Block section. The interrupt source can be configured through the Backup_clock_status_reg register (0x1C7). The related status is reported in the same register
- the signature unit of one of the two channels, in case of a wrong signature. The interrupt source can be configured through the Flash_enable registers (0x100 and 0x120). The related status is reported in the same.

6.15.5.4.1 Interrupt Register

This register latches:

- the status of all the interrupt request towards the external microcontroller
- the halt signal generated by the automatic diagnosis towards the four microcores

Table 100. Interrupt_status Register (0x1D4)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved			checksum_ch2	checksum_ch1	cksys_missing	SPI_irq	drv_irq	irq_uc1_ch2	irq_uc0_ch2	irq_uc1_ch1	irq_uc0_ch1	halt_uc1_ch2	halt_uc0_ch2	halt_uc1_ch1	halt_uc0_ch1
R/W	-			r	r	r	r	r	r	r	r	r	r	r	r	r
Lock	no			no	no	no	no	no	no	no	no	no	no	no	no	no
Reset	000			0	0	0	0	0	0	0	0	0	0	0	0	0

Table 101. Interrupt Register Bit Description

Bit Name	Function
halt_uc0_ch1	‘1’ if the automatic diagnosis has detected a short-circuit on uc0 ch1
halt_uc1_ch1	‘1’ if the automatic diagnosis has detected a short-circuit on uc1 ch1
halt_uc0_ch2	‘1’ if the automatic diagnosis has detected a short-circuit on uc0 ch2
halt_uc1_ch2	‘1’ if the automatic diagnosis has detected a short-circuit on uc1 ch2
irq_uc0_ch1	‘1’ if the microcode of uc0 ch1 has asserted its interrupt request
irq_uc1_ch1	‘1’ if the microcode of uc1 ch1 has asserted its interrupt request
irq_uc0_ch2	‘1’ if the microcode of uc0 ch2 has asserted its interrupt request
irq_uc1_ch2	‘1’ if the microcode of uc1 ch2 has asserted its interrupt request
drv_irq	‘1’ if the driver status block has disabled the output drivers
SPI_irq	‘1’ if the SPI interface has detected an error on the SPI communication

Table 101. Interrupt Register Bit Description (continued)

cksys_missing	‘1’ if the clock monitor has detected a cksys missing condition
checksum_ch1	‘1’ if the checksum of the code RAM of ch1 is wrong
checksum_ch2	‘1’ if the checksum of the code RAM of ch2 is wrong

The Figure 27 provides an overview of way configure and handle the interrupts according to the registers.

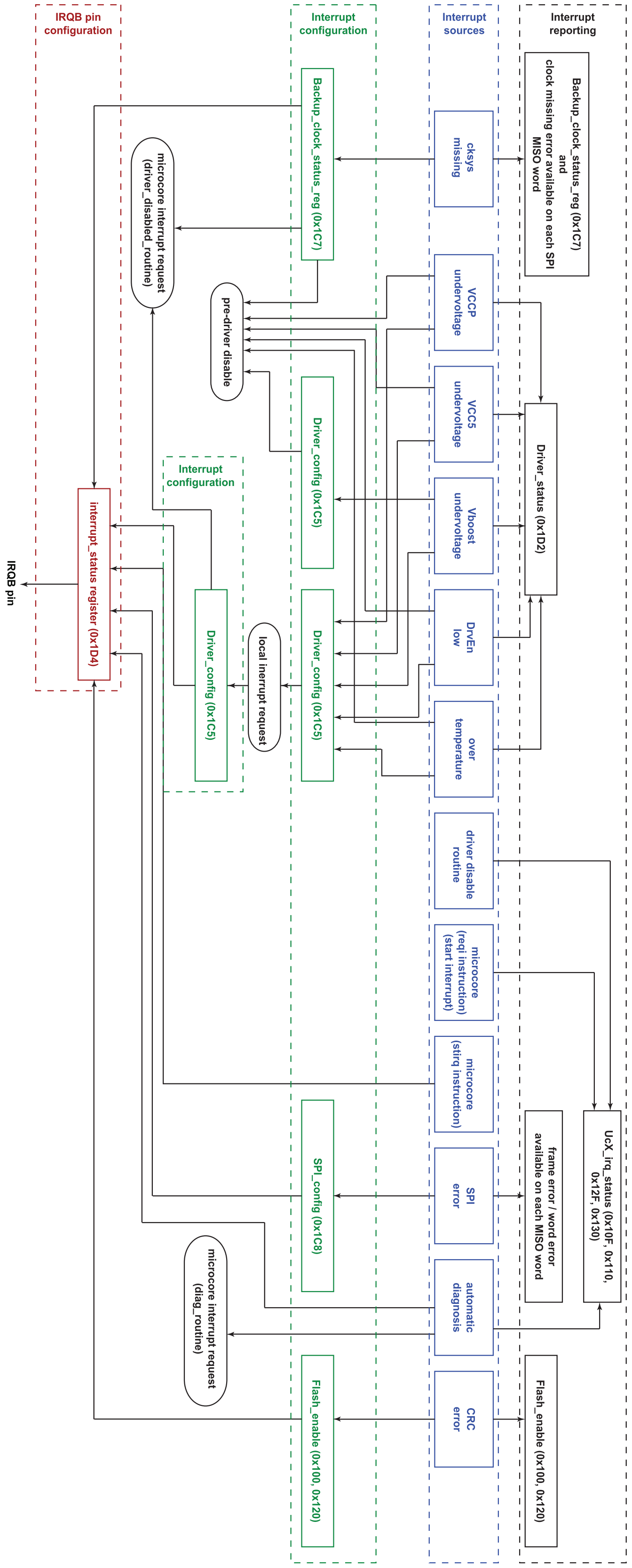


Figure 27. Interrupts Sources Configuration and Handling Overview

6.15.5.5 Backup_clock_status_handle Block

This block contains an 8-bit register (Backup_clock_status_reg register) that handles the loss of its input clock reference.

Table 102. Backup_clock_status_reg Register (0x1C7)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	-							timing_ violation	cksys_ missing_ disable_ driver	uc1_ ch2_ irq_en	uc0_ ch2_ irq_en	uc1_ ch1_ irq_en	uc0_ ch1_ irq_en	mcu_ irq_en	switch _to_ clock_ pin	loss_ of_ clock
R/W	-							r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r
Lock	-							no	yes	yes	yes	yes	yes	yes	no	no
Reset	-							0	0	0	0	0	0	0	0	0

The 8-bit register provides clock signal status monitoring to the external microcontroller. This is done by latching the information where a switch to the backup_reference signal has occurred:

- timing_violation: this bit is set if a timing violation has been detected. Writing a ‘1’ value on this register bit, resets the timing violation information.
- loss_of_clock: this read-only bit (loss_of_clock) latches the condition when the input reference is missing. These conditions are described in the PLL and Backup Clock section. The loss_of_clock bit can be reset by applying a valid clock frequency to the CLK device pin and setting the switch to clock pin bit high.
- switch_to_clock_pin: this bit (active on rising edge) provides a way to reset the loss of clock condition. If this bit is set during a loss of clock condition, it is reset as soon as the clock manager switches the PLL input to the external reference. If this bit is set while there is no loss of clock, the bit is reset immediately without any effect.
- mcu_irq_en: this bit generates an interrupt request to the microcontroller when cksys missing is detected. This interrupt is active until this register is read.
- uc0_ch1_irq_en: this bit enables the generation of an interrupt request to microcore 0 channel 1 when cksys missing is detected.
- uc1_ch1_irq_en: this bit enables the generation of an interrupt request to microcore 1 channel 1 when cksys missing is detected.
- uc0_ch2_irq_en: this bit enables the generation of an interrupt request to microcore 0 channel 2 when cksys missing is detected.
- uc1_ch2_irq_en: this bit enables the generation of an interrupt request to microcore 1 channel 2 when cksys missing is detected.
- cksys_missing_disable_driver: if this bit is set, the output drivers are disabled via the signal cksys_drven, as long as the cksys_missing signal is ‘1’, for a typical duration of 25 μs, as defined by t_{PLL_RELOCK}.

The interrupt to the external microcontroller and to the microcores is triggered as long as the cksys_missing signal is set. The microcore is able to process the interrupt as soon as there is a valid clock signal available on the CLK pin. The interrupt is triggered every time the Clock_manager switches to the internal clock reference, and when the clock manager tries to switch back to the external clock reference as this action is requested via the SPI.

6.15.5.6 Cipher_unit Block

This block has the function to secure the code downloaded by the microcontroller into the Code RAM via the SPI. The data loaded at device startup must be encrypted with the suitable cipher. This block receives an encoded SPI stream and decodes it at runtime. The decoded microcode is then stored in the Code RAM. This feature cannot be disabled.

The cipher algorithm is re-initialized every time the code memory is selected by a write operation to the Selection register (0x3FF).

6.15.5.7 Communication_interface Block

The access to memories and register is managed, paging the addresses through a SPI accessible register, named Selection_reg.

6.15.5.7.1 Selection_reg Register

The selection register is a 3-bit register aimed to select before starting the read/write operations toward a given address, which internal Code RAM is accessed, or to select all the other addresses, including the two Data RAMs and all the registers.

Table 103. Selection_reg Register (0x3FF)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved													comm_page_sel	CRAM_ch2_sel	CRAM_ch1_sel
R/W	-													r/w	r/w	r/w
Lock	-													no	no	no
Reset	0000000000000													0	0	0

Table 104 details the meaning of the three bits in this register. Some bit combinations are not allowed in this register and is ignored.

Table 104. Selection Register

Selection Register (MSB to LSB)	Enablement Address
000	No page selected. Further SPI operation is ignored, except for the one concerning this register.
001	Channel 1 Code RAM selected.
010	Channel 2 Code RAM selected.
011	Write operation affects both channel’s Code RAM. Read operation is not possible in this case.
100	Common page selected
101	The two LSB are ignored. “100” is written to the register. Common page selected.
110	
111	

This selection register is a unique register that is accessed from the SPI, whatever the value of the selection register. The two Code RAMs can be written in parallel during Normal mode.

Based on the values of the address bus (a_bus) and the selection register, the communication_interface block generates the address map of the device. This is achieved through the following signals working as chip selects for all the addressing areas of the component as detailed in Table 105.

Table 105. Map Areas Selection

Chip Select	Area Addressed
ch_sel_1(0)	Data RAM of channel 1
ch_sel_1(1)	Code RAM of channel 1
ch_sel_1(2)	Configuration registers of channel 1
ch_sel_2(0)	Data RAM of channel 2
ch_sel_2(1)	Code RAM of channel 2
ch_sel_2(2)	Configuration registers of channel 2
ext_sel_mcr	Main configuration register: all the generic registers located in the channel_management block
ext_sel_io	I/O configuration registers
ext_sel_diag	Diagnosis configuration registers

6.15.5.8 Prescalers Block

This block contains all the clock dividers available in the 33816. The divider ratios can be set in the Ck_pre register via the SPI.

6.15.5.8.1 Clock Prescaler

This 6-bit register sets the divider ratio to generate the ck clock, based on the cksys clock signal. This clock feeds the two act_channel blocks registers, except those accessible with the SPI, which directly clocked by cksys.

$f_{CK} = f_{CKSYS}/(ck\_per + 1) = 1/t_{CK}$

Note that the actual divider ratio is ck_per + 1. Setting ck_per to “000100” ck is cksys/5.

Table 106. Ck_per Register (0x1C0)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										ck_per					
R/W	-										r/w					
Lock	-										yes					
Reset	0000000000										000000					

The different device/channel operating mode available according to the Ck_per setting are described in Table 107.

Table 107. Ck_per And Device Modes

Ck_per	Clock divider	Read And Write SPI Access To Registers And DRAM	Single Microcore	Dual Microcore	Drive Outputs From Flag Pins
0	1	yes	no	no	no
1	2	yes	yes	no	yes
2	3	yes	yes	no	yes
≥3	≥4	yes	yes	yes	yes

6.15.5.8.2 Clock Offset Compensation Prescaler

This 8-bit register sets the divider ratio to generate the ck_ofscmp clock, based on the cksys clock signal. This clock feeds the offset recovery counters of the current measure interface.

Note that the actual divider ratio is ck_ofscmp_per + 1. Setting ck_ofscmp_per to “00001000” ck_ofscmp is cksys/9.

Table 108. Ck_ofscmp_per Register (0x1C4)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								ck_ofscmp_per							
R/W	-								r/w							
Lock	-								yes							
Reset	00000000								0000000							

6.15.5.9 Driver_enable Block

This block generates the enable signal to the output drivers (signal en_hs for the high-side drivers, and en_ls for the low-side drivers). The conditions described below must be considered to enable the drivers. Some of them are mandatory and others are configurable through the Backup_clock_status_reg (0x1C7) and the Driver_config (0x1C5) registers.

The low-side one to six output drivers are enabled when:

- cksys_drven = 1 – The input clock signal is not missing (configurable)
- drv_en = 1 – The DRVEN pin is high (mandatory)
- uv_vccp = 0 – There is not undervoltage on V_{CCP} (mandatory)
- uv_vcc5 = 0 – There is not undervoltage on V_{CC5} (mandatory)

The low-side seven output driver is enabled when:

- cksys_drven = 1 – The input clock signal is not missing (configurable)
- drv_en = 1 – The DRVEN pin is high (configurable)
- uv_vccp = 0 – There is not undervoltage on V_{CCP} (mandatory)
- uv_vcc5 = 0 – There is not undervoltage on V_{CC5} (mandatory)

The high-side output drivers are enabled when:

- cksys_drven = 1 – The input clock signal is not missing (configurable)
- drv_en = 1 – The DRVEN pin is high (mandatory)
- uv_vccp = 0 – There is not undervoltage on V_{CCP} (mandatory)
- uv_vcc5 = 0 – There is not undervoltage on V_{CC5} (mandatory)
- uv_vboost = 0 - There is not undervoltage on V_{BOOST} (configurable)

6.15.5.9.1 Driver Status

This 7-bit register provides a monitoring of the output drivers status to the external microcontroller. This is done by latching any error condition which disables the output drivers. Some of these error conditions must be enabled through the Backup_clock_status_reg (0x1C7) and the Driver_config registers (0x1C5).

Table 109. Driver_status Register (0x1D2)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved									cksys_missing	DrvEn_latch	DrvEn_value	over_temp	uv_boost	uv_vcc5	uv_vccp
R/W	-									r	r	r	r	r	r	r
Lock	-									no	no	no	no	no	no	no
Reset on read										conf.	conf.	no	conf.	conf.	conf.	conf.
Reset	0000000									0	1	-	0	0	0	0

cksys_missing: this bit is set if the cksys missing condition of the clock_manager block disables the drivers. This condition can be configured in the Backup_clock_status_reg register (0x1C7).

DrvEn_latch: this bit latches the condition when the DRVEN input pin is inactive.

- 1: DRVEN pin was NOT low since last reset of the driver_status register
- 0: DRVEN pin was low since the last reset of the driver_status register

DrvEn_value: this bit is not an error condition but only a “living copy” of the DRVEN pin.

- 1: DRVEN pin is high
- 0: DRVEN pin is low

over_temp: this bit latches the condition that an overtemperature is present. It is not used to disable the drivers.

uv_vboost: this bit is set if the undervoltage on the vboost disables the high-side drivers. This condition can be configured through the Driver_config register (0x1C5).

uv_vcc5: this bit latches the undervoltage condition on V_{CC5}.

uv_vccp: this bit latches the undervoltage condition on V_{CCP} or the error issued from GND loss detection.

Once an error bit has been set, it can only be reset by a SPI write operation in this register, considering that the corresponding error is no longer present. The same error bits are reset even upon SPI read operations, but only when a proper enable bit is set in the Reset_behavior register (0x1CE).

6.15.5.9.2 Driver Configuration

This register allows to configure the conditions leading to the driver disable and to the interrupt generation. The interrupt request generation towards the microcontroller and the microcores can also be set.

Table 110. Driver_config Register (0x1C5)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	hs5_ ls36_ ovr	vccp_ ext_en	ls7_ ovr	vboost _mon_ en	vboost _disabl e_en	over_ temp_ irq_en	drv_en _irq_ en	vboost _irq_ en	vcc5_ irq_en	vccp_ irq_en	iret_en	irq_ uc1 ch2 en	irq_ uc0_ ch2_ en	irq_ uc1_ ch1_ en	irq_ uc0_ ch1_ en	irq_ mcu_ en
R/W	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Reset	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

The driver_enable block generates a local interrupt masking the five possible disable conditions through the bits detailed below:

- drv_en_irq_en: if set, the drv_en generates the local interrupt.
- vboost_irq_en: if set, an undervoltage on V_{BOOST} generates the local interrupt.
- vcc5_irq_en: if set, an undervoltage on V_{CC5} generates the local interrupt.
- vccp_irq_en: if set, an undervoltage on V_{CCP} generates the local interrupt.
- over_temp_irq_en: if set, the overtemperature condition generates the local interrupt.

If a local interrupt is generated, it is possible to propagate it to an external device (microcontroller) and to the four microcores. This is done when the following bits are set:

- irq_mcu_en, for the external device through the IRQB pin
- irq_uc0_ch1_en, for the microcore 0 of channel 1
- irq_uc1_ch1_en, for the microcore 1 of channel 1
- irq_uc0_ch2_en, for the microcore 0 of channel 2
- irq_uc1 ch2 en, for the microcore 1 of channel 2

This register contains some other configuration bit related to the output drivers:

- iret_en: the driver_enable block automatically generates a return from interrupt request towards all the microcores. This request can be filtered by microcode if not required. Two kind of return from interrupt is selectable. If iret_en is set to ‘0’, a return from interrupt request is sent to the microcores when the drivers are re-enabled after a disable condition. If iret en is set to ‘1’, a return from interrupt request is sent to the microcores when the Drivers_status register (0x1D2) is cleared. For the return from interrupt to happen the driver status register must be write or read while the reset on read configuration is activated in the Reset_behavior register (0x1CE)
- vboost_disable_en: if set, an undervoltage of V_{BOOST} disables the output drivers.
- vboost_mon_en: this signal configures the divider on the V_{BOOST} voltage. If vboost_mon_en is set to ‘0’, V_{BOOST} is divided by 32 and then compared with a threshold. If vboost_mon_en is set to ‘1’, V_{BOOST} is divided by 4 and then compared with a threshold.
- ls7_ovr: if set to ‘1’, the low-side seven output driver is not influenced by the DrvEn signal.
- hs5_ls36_ovr: if set to ‘1’, the high-side five and low-side three and six output driver is not influenced by the DrvEn signal.
- vccp_ext_en: if set to ‘0’, the internal voltage regulator is enabled and the corresponding pin is used only to connect a bypass capacitor. If set to ‘1’, the internal voltage regulator is disabled and the V_{CCP} voltage must be supplied externally through the corresponding pin. During bootstrap switch init (refer to Bootstrap switch control section) this setting is bypassed and the value of the vccp_ext_enable signal is set to the inverted value of the DBG pin sampled at reset (POResetB and ResetB) (see Table 111 for more details). In this case, the DBG pin, at reset, needs to be configured as an input, whose value is latched at the rising edge of the POResetB and ResetB signal, and used to set the configuration of the V_{CCP} internal regulator during the init phase of the

bootstrap switch. A SPI reset leaves the latched information unchanged. The DBG pin has an internal weak pull-up resistor so its value is ‘1’ when not connected (n. c.).

The different device/channel operating mode available according to the Ck_per setting are described in Table 111.

Table 111. V_{CCP} External Enable Setting

DBG Pin (latched)	SPI Bit	Booststrap Init (Min. 1 HS)	V_{CCP} External Supply Enablement
1 (n.c.)	-	1	0 (Internal regulator)
1 (n.c.)	1	1	1 (External V_{CCP} supply)
1 (n.c.)	0	0	0 (Internal regulator)
0	1	-	1 (External V_{CCP} supply)
0	0	-	0 (Internal regulator)

6.15.5.10 SPI_access_controller Block

All the SPI accessible registers can be accessed also by the microcores through a “SPI backdoor”. However, Data and Code RAMs are unavailable through the backdoor. The SPI_access_controller block receives all the register read/write requests, from the SPI interface and from all the enabled microcores.

The requests coming from the SPI interface are considered with the highest priority. When these requests are received, the requested operation (register read/write) is immediately performed.

All the requests from the microcores are performed before the end of the next ck clock cycle (the clock cycle used for the microcode execution). It means that the result of the operation is available for the second instruction after the read/write request. The value of the SPI backdoor registers must not be changed until the backdoor operation is finished.

6.15.5.11 Trace_unit Block

The step by step evolution of the code execution for all the microcores can be traced throughout the DBG pin. This allows obtaining the microprogram counter values of the microcores in real time: the microprogram counter value (uPC) corresponds to the address of the instruction being executed.

To allow tracing, the DBG pin is used as an asynchronous serial line running at the cksys clock frequency. The device translates the data into microprogram counter values. The value is transmitted back, thanks to the tracer function through the DBG pin.

If the clock prescaler (refer to Ck_per (0x1C0) section for more details) is set to three, full trace mode is enabled whatever the value of PLL_factor bit in the register (PLL_config (0x1C6). For all other clock prescaler values, the trace mode doesn’t work properly (clock prescaler > 3) or is not available (clock prescaler < 3). The MSB is always transmitted first on the DBG pin.

6.15.5.11.1 Full Trace Mode

Considering a point in the code execution, there are a very limited number of possibilities for the next values of the microprogram counter. Trace is then implemented, such as to transmit a code on four bits issued for the microprogram counter values.

The trace operation consists of reconstructing the execution path from the codes that the device transmits to the tracer. However, these codes only describe variation of the uPC value. To obtain the actual execution path, the trace operation must start from a point in the code known by the tracer. From the starting point and knowing all the variations, it is possible to obtain the uPC path.

The trace can be activated on one microcore at a time.

The trace sequence is composed of five steps:

1. Calibrate: The communication between the 33816 and the tracer is asynchronous, since no clock line is shared. The first frame transmitted through the DBG pin is a burst of eight clock cycles with a frequency half of the internal cksys.

At the end of this phase the DBG pin stays low for at least one ck clock period.

2. Sync: The sync point is specified in Trace_start register (0x1CA). If the trace operation is enabled in the Trace_config register (0x1CC) and the trace unit is in idle state, when the uPC value of the selected microcore reaches the sync point, the 4 bits code “1010” (start sync) is transmitted on the DBG pin.

Then the trace_unit goes to the next phase.

3. Trace: Each cycle is transmitted a four bit code value that identifies which path has been taken by the code execution among possible ones. Due to the many possible paths, to keep the number of codes to 14 (‘0000’ and ‘1111’ are not used because they are difficult to handle in an asynchronous communication), some codes have different meaning according to which instruction is being executed. Some codes have different meaning if you are executing normal code or if you are inside an interrupt service routine (ISR).

6.15.5.11.2 Normal Execution Trace

The codes used during normal execution are the following:

- Code “0101”, default path taken. It means that the path taken after the current instruction is the default one. For nearly all instructions, it means that the instruction has not altered the code flow, no interrupt has been received, so the following instruction is the next one. However their are some exceptions:
    - Unconditional jumps (*jmpr, jmpf*) and software interrupt requests (*reqi*) cannot produce this code.
    - This code is produced by the *wait* instruction if the wait is fulfilled and the wait entry 1 is selected as next destination.
- Code “1010”, forked path taken. It means that the instruction has altered the code execution path. It also means that no hardware interrupt has been received. The new uprogram counter value depends on the exact instruction that is currently under execution. For this reason, the tracer device must be provided with the microcode, to correctly select the new uPC value.

Only some instruction can produce this code:

- The *wait* instruction produces this code when the uprogram counter is unchanged (the code is waiting).
- All the jump instructions produce this code when the jump is taken.
- The jump to subroutine instructions (*jtsr, jtsf*) always produce this code.
- The software interrupt request instruction (r*eqi*) always produces this code.

- Code “0100”, automatic diagnosis interrupt. It means that the code execution has been interrupted for a fault detected by the automatic diagnosis. The new uprogram counter value is the start of the diagnosis interrupt routine. The tracer must be aware of the interrupt table to correctly select the new uPC value.
- Code “0010”, driver disabled interrupt. It means that the code execution has been interrupted for a fault that leads to disable the output drivers.

The new uprogram counter value is the start of the driver disabled interrupt routine. The tracer must be aware of the interrupt table to correctly select the new uPC value.

- Code “0011”, start edge interrupt. It means that the code execution has been interrupted by a software interrupt caused by a start edge (Refer to Sw_interrupt_routine_addr (0x10E, 0x12E) for configuration). The new uprogram counter value is the start of the software interrupt routine. The tracer must be aware of the interrupt table to correctly select the new uPC value.
- Code “0110”, wait entry 2 selected. This code can be produced only by the wait instruction (wait) if the wait is fulfilled and the wait entry 2 is selected as next destination.
- Code “1011”, wait entry 3 selected. This code can be produced only by the wait instruction (wait) if the wait is fulfilled and the wait entry 3 is selected as next destination.
- Code “1101”, wait entry 4 selected. This code can be produced only by the wait instruction (wait) if the wait is fulfilled and the wait entry 4 is selected as next destination.
- Code “1001”, wait entry 5 selected. This code can be produced only by the wait instruction (wait) if the wait is fulfilled and the wait entry 5 is selected as next destination.

6.15.5.11.3 Interrupt Execution Trace

The most difficult concept is how to trace the flow of the uPC after the exit from an interrupt routine.

If execution restarts, the next uPC value is the microcore entry point (refer to Uc0_entry_point (0x10A, 0x12A) and Uc1_entry_point (0x10B, 0x12B) sections for entry point configuration).

If execution continues, the last code of the ISR point to which direction the flow continues, referring to the last instruction executed before the ISR was called. The tracing is ‘broken’ if the sync point is inside the ISR routine.

If the execution restarts, after the ISR or the sync point is during normal code or inside another ISR, trace is not limited.

Interrupt return (iret) instruction called with the “restart” parameter produces a fixed code (“1010”).

Interrupt return (iret) instruction called with the “continue” parameter can produce any of the codes used during normal execution, except the codes referring to interrupt requests (“0010”, “0011” and “0100”). Using the same rules, this code point to the path selected by the last instruction before the ISR.

Automatic interrupt return request have the same possible destinations as the interrupt return (iret) instruction called with the “continue” parameter. However in this case, the codes are different from the ones used during normal execution.

- Code “0101”, default path taken. It means that the path taken after the current instruction is the default one. For nearly all instructions it means that the instruction has not altered the code flow, no interrupt has been received, so the following instruction is the next one.

The are some exceptions:

  - Unconditional jumps (*jmpr, jmpf)* cannot produce this code.
  - Interrupt return (*ret*) instruction produces this code when they have the “continue” parameter and the path to be taken after the ISR is the default one (includes wait entry 1 selected).

This code is produced by the wait instruction (wait) if the wait is fulfilled and the wait entry 1 is selected as next destination.

- Code “1010”, forked path taken. It means that the instruction has altered the code execution path. It also means that no hardware interrupt has been received. The new microprogram counter value depends on the exact instruction that is currently under execution. For this reason, the tracer device must be provided with the microcode to correctly select the new uPC value.

Only some instruction can produce this code:

  - Interrupt return (*iret*) instruction produces this code when they have the “restart” parameter.
  - Interrupt return (*iret*) instruction produces this code when they have the “continue” parameter and the path to be taken after the ISR is the forked one.
  - The *wait* instruction produces this code when the uprogram counter is unchanged (the code is waiting).
  - All the jump instructions produce this code when the jump is taken.
  - The jump to subroutine instructions (*jtsr, jtsf*) always produce this code.

- Code “0110”, wait entry 2 selected. This code can be produced by the wait instruction (*wait*) if the wait is fulfilled and the wait entry 2 is selected as next destination. It is also produced by the interrupt return (*iret*) instruction when they have the “continue” parameter, a wait instruction was executing when ISR was called and the destination 2 was about to be selected.
- Code “1011”, wait entry 3 selected. This code can be produced by the wait instruction (*wait*) if the wait is fulfilled and the wait entry 3 is selected as next destination. It is also produced by the interrupt return (*iret*) instruction when they have the “continue” parameter, a wait instruction was executing when ISR was called and the destination 3 was about to be selected.
- Code “1101”, wait entry 4 selected. This code can be produced by the wait instruction (*wait*) if the wait is fulfilled and the wait entry 4 is selected as next destination. It is also produced by the interrupt return (*iret*) instruction when they have the “continue” parameter, a wait instruction was executing when ISR was called and the destination 4 was about to be selected.
- Code “1001”, wait entry 5 selected. This code can be produced by the wait instruction (*wait*) if the wait is fulfilled and the wait entry 5 is selected as next destination. It is also produced by the interrupt return (*iret*) instruction when they have the “continue” parameter, a wait instruction was executing when ISR was called and the destination 5 was about to be selected.
- Code “0001”, forked path after automatic interrupt return. This code can be produce only when an automatic interrupt return is received and the path to be taken after the ISR is the forked one.
- Code “0111”, default path after automatic interrupt return. This code can be produced when an automatic interrupt return is received and the path to be taken after the ISR is the default one (includes wait entry 1 selected).
- Code “0100”, wait entry 2 selected after automatic interrupt return. This code can be produce only when an automatic interrupt return is received, a wait instruction was executing when ISR was called and the destination 2 was about to be selected.
- Code “0010”, wait entry 3 selected after automatic interrupt return. This code can be produce only when an automatic interrupt return is received, a wait instruction was executing when ISR was called and the destination 3 was about to be selected.
- Code “1110”, wait entry 4 selected after automatic interrupt return. This code can be produce only when an automatic interrupt return is received, a wait instruction was executing when ISR was called and the destination 4 was about to be selected.
- Code “0011”, wait entry 5 selected after automatic interrupt return. This code can be produce only when an automatic interrupt return is received, a wait instruction was executing when ISR was called and the destination 5 was about to be selected.

1. Stop Sync: The trace operation is not meant to last indefinitely. It is possible to define a “stop” address (refer to the Trace_stop register (0x1CB) to define stop address). If during the precedent phase (trace) the uPC reaches the stop address, the code “1100” is sent on the DBG pin and the trace_unit goes to the following phase.

2. Post Trigger: The trace operation continues for a fixed number of ck clock cycles. After this time has elapsed, the trace_unit goes to idle state.

6.15.5.11.4 Trace_start

Table 112. Trace_start Register (0x1CA)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved						start_address									
R/W	-						r/w									
Lock	-						no									
Reset	000000						00000000000									

The trigger_address field contains the address that is used to synchronize the 33816 trace_unit with the external tracer. If the trace operation is enabled (prefer to Trace_config register (0x1CC) for trace unit enablement) and the trace unit is in idle state, when the uPC value of the selected microcore reaches this address, the sync code is transmitted on the DBG pin. The trace_unit then goes to the next phase.

6.15.5.11.5 Trace_stop

Table 113. Trace_stop Register (0x1CB)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved						stop_address									
R/W	-						r/w									
Lock	-						no									
Reset	0						0									

The stop_address field contains the address that is used to finalize the trace operation. If the trace operation is ongoing (trace phase), when the uPC value of the selected microcore reaches this address, the stop code is transmitted on the DBG pin. Then the trace_unit goes to the next phase (post trigger phase).

6.15.5.11.6 Trace_config

Table 114. Trace_config Register (0x1CC)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved					Trace enable	uc_select		post_trigger_length							
R/W	-					r/w	r/w		r/w							
Lock	-					no	no		no							
Reset	00000					0	00		00000000							

- trace_enable. When this bit is set to ‘1’, the trace_unit start the first phase of the trace operation. This bit can be set to ‘0’ by the user, to immediately stop the device trace unit transmission. This bit is automatically reset after the trace operation is complete.
- uc_select. Select which is the microcore target of the trace operation:
    - “00”: microcore 0, channel 1.
    - “01”: microcore 1, channel 1.
    - “10”: microcore 0, channel 2.
    - “11”: microcore 1, channel 2.
- post_trigger_length. This field selects the duration of the post trigger phase, expressed as number of ck clock cycles. However, setting the post_trigger_length field to 255 generates a continuous stream after the stop point. The trace_unit can be simply deactivated by writing “0” in the trace_enable bit.

6.15.5.12 SPI_interface_slave Block

'The communication between the 33816 and the main microcontroller is managed with a 16-bit SPI interface. This block includes four sub-blocks meant to control the SPI transfer:

- spi_interface_slave
- spi_config
- spi_protocol
- spi_error

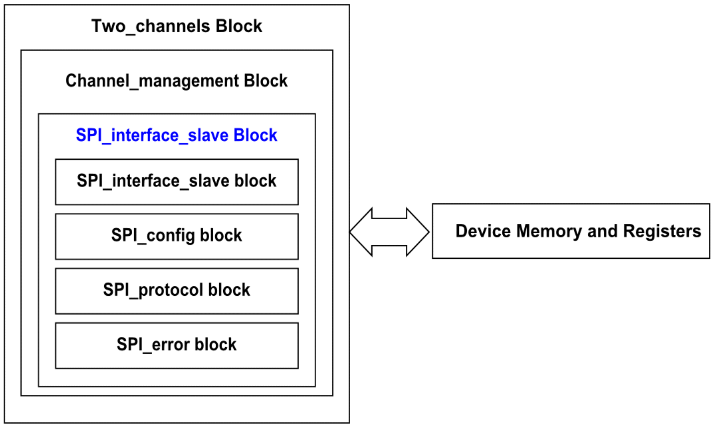


Figure 28. Spi_interface_slave Block Diagram

6.15.5.12.1 Spi_interface_slave Block

This block is the module providing the SPI connection features. It only works as a slave SPI module, allowing only 1- bit data transactions. The device requires a cphase value of 1 and a cpol value of 0. This means that the SPI module samples the MOSI signal, during write operations on the falling edge of the serial clock sclk. Likewise, during read operations, the SPI module always puts the output value available on the MISO signal on the rising edge of the sclk clock. The cpol value of ‘1’ can be implemented by adding an external inverter.

6.15.5.12.2 Spi_config Block

The spi_config register is an 8-bit register storing the SPI protocol configuration and SPI diagnosis.

Table 115. Spi_config Register (0x1C8)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								miso_ slewrate	protocol_ mode	irq_en	watchdog				
R/W	-								r/w	r/w	r/w	r/w				
Lock	-								yes	yes	yes	yes				
Reset	0								0	0	0	01010				

- miso_slewrate: selects one of the two possible values for the slew rate of the MISO pin.
- protocol_mode: select the type of burst transmission accepted by the protocol, ‘0’ means mode A, ‘1’ means mode B.
- irq_en: enable the SPI interface to request an interrupt towards the microcontroller, in case an incorrect SPI transmission is received.
- watchdog: when using mode A, the maximum time the SPI chip select can be inactive during a burst is expressed as follows:
  - $t_{WATCHDOG} = t_{CKSYS}$ ((watchdog +1) * 32768)

where t_{CKSYS} is the period of the cksys internal clock.

The SPI protocol mode can be selected through the SPI. In this case, the SPI transmission must be compatible to mode A and B (see SPI read access and Mode B section for A and B compatible protocol description). The number of operations for the SPI transmission cannot be ‘0’ and the chip select must not be deassert during the transmission.

The protocol mode can be changed at any time.

6.15.5.13 Spi_protocol Block

The spi_protocol block allows managing the location where the incoming data from the SPI_interface is stored. It also routes the path that is used to access data when a SPI reading is requested. After reset and after SPI transmission is completed, the protocol always waits for the16-bit control word.

Table 116. SPI Control Word Description

Control Word Area	Description
Control_word [15]	r_w: read (1)/write (0) operations
Control_word [14:5]	offset: start address
Control_word [4:0]	number: number of operations

The field ‘r_w’ defines if the action is to read data (r_w = ‘1’) through the SPI or write (r_w = ‘0’) incoming data in registers.

The field ‘number’ defines the number of 16-bit words read or written by the external microcontroller. The offset value is limited to a maximum of 31 words.

The field ‘offset’ defines where the read or write operation must start so what the first address in this burst of operations to be accessed.

To detect corrupted burst of data, the protocol monitors the burst, according to the ‘protocol mode’ bit of the SPI_config Register (0x1C8).

6.15.5.13.1 SPI Read Access

A SPI frame for read access consists of 2 to 32 16-bit words. The way the number of words is defined depends on the SPI mode used (A or B). The table below shows the data transmitted on MOSI and MISO. During the command word the check byte and the SPI error status is transmitted via the MISO line.

The MSB is always transmitted first.

Table 117. SPI Read Access

MOSI word 1 (control word) – Read access																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	r_w	offset										number				
Value	1	0 to 1023										n=0 to 31				
MISO word 1 (control word) – Read access																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	check byte													cksys missing	frame error	word error
Value	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0
MOSI word 2 (data) – Read access																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	(empty)															
Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 117. SPI Read Access (continued)

MISO word 2 (data) – Read access																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	read data 1															
Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
MOSI word n+1 (data) – Read access																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	(empty)															
Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MISO word n+1 (data) – Read access																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	read data n															
Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

6.15.5.13.2 SPI write access

A SPI frame for write access consists of 2 to 32 16-bit words. The way the number of words is defined depends on the SPI mode used (A or B). the table below shows the data transmitted on MOSI and MISO. During the command word and all data words the check byte and the SPI error status is transmitted via the MISO line.

The MSB is always transmitted first.

Table 118. SPI Write Access

MOSI word 1 (control word) – Write access																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	r_w	offset										number				
Value	0	0 to 1023										n=0 to 31				
MISO word 1 (control word) – Write access																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	check byte													cksys missing	frame error	word error
Value	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0
MOSI word 2 (data) – Write access																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	write data 1															
Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MISO word 2 (data) – Write access																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	check byte													cksys missing	frame error	word error
Value	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0

Table 118. SPI Write Access (continued)

MOSI word n+1 (data) – Write access																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	write data n															
Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MISO word n+1 (data) – Write access																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	check byte													cksys missing	frame error	word error
Value	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0

6.15.5.14 Mode A

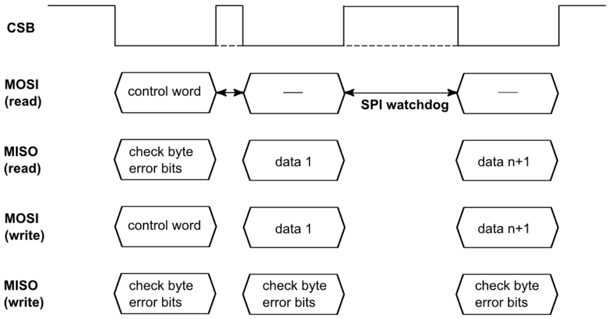


Figure 29. SPI Protocol Mode A

The maximum delay between data belonging to the same frame is specified by the spi_watchdog parameter bits of the Spi_config register (0x1C8). Between one data transfer and the next, the SPI chip select can be asserted on not. If this delay exceeds the watchdog time, the SPI interface goes into error state.

It is possible to perform long frame transfers by sending a control word with the parameters number and offset set to zero. The effect varies according to the current value of the selection register bits defined in the Selection_reg register (0x3FF):

- if the value of the select register is “001”, the protocol transmits a frame allowing operations starting from the address 0; the number of operations is specified by the value of the Code_width register (0x107) of channel 1. This command is used to write the whole Code RAM of channel one with only one command word.
- if the value of channel select register is “010”, the protocol transmits a frame allowing operations starting from the address 0; the number of operations is specified by the value of the Code_width register (0x127) of channel 2. This command is used to write the whole Code RAM of channel two with only one command word.
- if the value of channel select register is “011”, the protocol transmits a frame allowing operations starting from the address 0; the number of operations is specified by the value of the Code_width register (0x0107) of channel 1. This command is used to fully write the Code RAMs of both channel (with exactly the same code) with only one command word.
- if the value of channel select register is “100”, the protocol transmits a frame allowing operations starting from the address 0; the maximum number of operations is 128. This command is used to fully write the Data RAMs of both channels with only one command word.
- For all the other values of channel select register, the command is ignored.

Transmission of control word with the parameter number set to zero and the parameter offset greater than zero is not allowed. It leads to data corruption in the registers or Data RAM. A SPI write access example is provided below. In this example, the used is setting the ADC conversion register 1 and 2 (0x194 and 0x195).

- The first step consists in selecting the communication interface as target. The selection is done by writing the value 0x0004 at the Selection register address (0x3FF). The first 16-bit words to be sent is ‘0_1111111111_00001’ (0x07FE1). As the ASIC is in idle conditions, the incoming data is a command word: write operation is selected (as the MSB is ‘0’) starting from address 0x03FF (the ten offset bits) and one data word is sent in the next frame (‘00001’ written in the 5 LSB). The next incoming frame (the data word) is 0x0004. As the number of word expected arrives, the SPI block returns to idle state;
- The second step is writing the value of the two ADC conversion register one and two: the SPI block is expecting a command word. The correct data to send is “0_0110010100_00010”. write operation is selected (as the MSB is ‘0’) starting form address 0x0194 (‘0110010100’ written in the offset field) and two data words sent in the next frame (‘00010’ written in the 5 LSB). The next incoming data is written in the ADC conversion register one (0x0194) then in following data into the ADC conversion register two (0x195).

6.15.5.15 Mode B

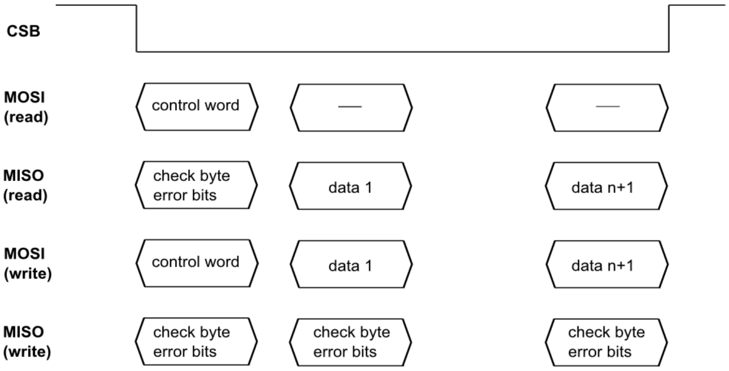


Figure 30. SPI Protocol Mode B

For all the duration of data transmission, the SPI chip select must be asserted. The first word after the chip select assertion is a command word and the following ones are the data word. If the number parameter is not zero, the SPI interface goes into an error state if:

- The chip select is de-asserted and the number of words transferred is lower than the number specified in the command word,
- The number of word transferred exceeding the number specified in the command word.

If the number parameter is zero, the number of data word transmitted is only determined by the assertion of SPI chip select.

Reading any register “reset on read” or any register located just before such a register using a mode B SPI communication with the number parameter set to zero is not recommended. It may lead to register reset even if no read out via the SPI. A such situation can be avoid by specifying the number data words transmitted in the command word.

6.15.5.15.1 Spi_error Block

Table 119. Spi_error Register (0x1D3)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved													cksys_ missing	frame_ error	word_ error
R/W	-													r	r	r
Lock	-													-	-	-
Reset	000000000000													0	0	0

The Spi_error register is a 3-bit register split as described by the following.

- cksys_missing: cksys missing error condition
- frame_error: frame incomplete error condition
- word_error: word incomplete error condition

The block monitors the spi_protocol and the spi_interface, and reports any errors during the communication with the microcontroller. If an error is detected, the corresponding code is stored in the SPI_error register. To warn the microcontroller, during the write transfer (from microcontroller to asic), the MISO signal transfers a diagnostic word:

- the first 13 bits of this word are constant (“1010101010101”) and are used to detect short circuits on the MISO line,
- the last three bits copy the three LSBs of the SPI_error register.

After an error code is written in this register, the register becomes write-protected to latch the error condition. In this case, any other error is ignored to avoid error cumulation effect issued from error source side effect.

In addition, an interrupt request can be generated towards the microcontroller, if the irq en bit is set to ‘1’ in the SPI_config register (0x1C8).

When an error is reported during SPI connection, the SPI protocol inside the 33816 moves to the error state. In this state, only a read access to the SPI_error register is allowed using the command word (0xBA61). A single word then transmitted to the SPI master and the error is reset (along with the interrupt if enabled).

If the value of the selection register was set to the value 0x04 (selecting the generic configuration registers) before the error, the word transmitted is the error code and the SPI_error register is immediately reset. Otherwise, a random word is sent and the error state of the SPI protocol is reset. In this second case, the error code register can be read (and thus reset) in a following frame. A detail of the error code is described by the following.

- cksys missing: this errors is set if an SPI transfer is required (the chip select csb signal is low) while the cksys clock is missing.
- frame error: this error is set if the number of data words in a burst is different from the one specified in the command word.
  - Mode A is selected, the slave_protocol block received a control word that specifies n word transfers, but the microcontroller performs less operations and then end the communication. In this case, this module provides a watchdog function: if during a programmed transfer, the communication with the microcontroller is inactive for a time longer than a prefixed limit, the transfer is considered aborted and an error is detected.
  - Mode B is selected, the number parameter is not zero in the command word and the number of transferred words is different from the one programmed in the command word.
- word error: during the transfer of a long word data frame the device received or sent an incorrect number of bit. If multiple words are being transferred in a row with the chip select always active (the fastest way), the error is detected at the end of the sequence and it is not possible to identify the incorrect word. To identify the incorrect data, the chip select must be deactivated and reactivated between each word transfer.

During normal operations, the SPI_error register value is 0x0000.

6.15.6 Act_channel Block

This block named act_channel is defined to operate one or two microcores, depending on the dual microcore mode enablement.

Two act_channel blocks are implemented in the device. Each microcore can be enable to all six start signals (refer to the Start_config_reg (0x104 and 0x124) section for more details). Each actuator can be controlled by a programmable number of output stages, normally including one high-side driver, one low-side driver and one optional freewheeling driver. The assignment of high-side and low-side output drivers to each actuator is flexible and can vary depending on the target application.

Virtually each microcore is able to control all the output drivers (globally five high-side and seven low-side are available on the device). In the application, the drivers are assigned through the configuration registers.

The Out_acc_ucX_chY (0x184, 0x185, 0x186, 0x187) configuration registers allow the access to any output driver by any microcore through microcode programming.

This block includes two programmable microcores Uc0 and Uc1, sharing the same data memory Data RAM and the same code memory Code RAM.

The second microcore of each channel can operate only if the following conditions are met:

- The clock prescaler (refer to the Ck_per register (0x1C0) section for more details) is set to a value greater or equal to three. In this condition, the internal ck clock period is at least four times the cksys clock period.
- The dual sequencing mode is enabled (refer to section Flash_enable (0x100 and 0x120)).

Each microcore controls a dedicated set of outputs (output drivers commands, DAC commands, Vds_threshold control and diagnosis, and OPAMP gain selection) which is combined with the same set of outputs coming from the other microcores.

The not-locked microcores are in a safe state because either they have still to be enabled, or they have been unlocked by the signature unit. DAC commands, Vds_threshold control and diagnosis, and operational amplifier gain selection are not affected, so they keep their former value (the reset value if all the microcores have still to be enabled), while the output drivers commands are all driven turned off. The turn off polarity can be specified in the output configuration registers (refer to LSx output register (0x140 to 0x151) and HSx output register (0x153 to 0x161) sections).

This architecture has been selected in order to have two concurrent microcores able to control overlapped actuations on two different loads, without having to provide two different Code RAMs.

This block integrates the following blocks:

- Parameters
- Dual_microcore_arbiter
- Code_RAM
- Signature_unit
- Micro_interface
- Ch_microcore (x2)

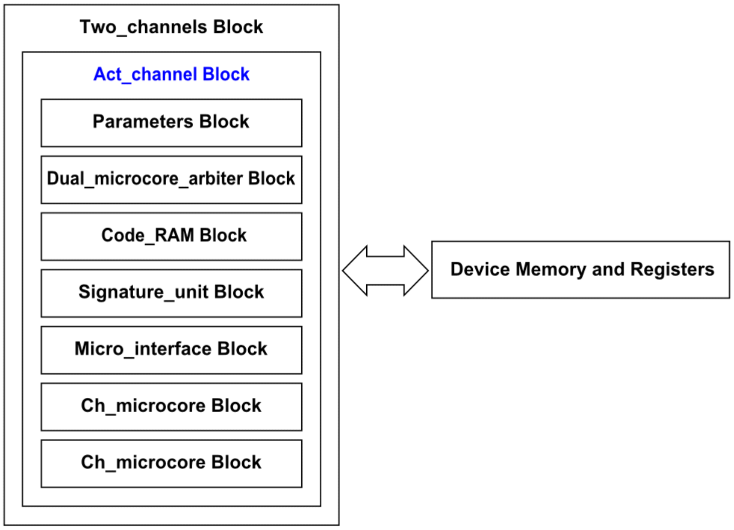


Figure 31. Act_channels Block Diagram

6.15.6.1 Parameters Block

This block contains all the configuration registers dedicated per channel:

- Flash Enable (flash_enable)
- Control Register Microcore 0 (ctrl_reg_uc0)
- Control Register Microcore 1 (ctrl_reg_uc1)
- Unlock Register (unlock_reg)
- Start Config Register (start_config_reg)
- Status Register Microcore 0 (status_reg_uc0)
- Status Register Microcore 1 (status_reg_uc1)
- Code Width Register (code_width)
- Checksum Register 16 MSBs (checksum_h)
- Checksum Register 16 LSBs (checksum_l)
- Entry Point Microcore 0 (uc0_entry_point)
- Entry Point Microcore 1 (uc1_entry_point)
- Diagnosis routine address (diag_routine_addr)
- Driver disabled routine address (driver_routine_addr)

- Software interrupt routine address (sw_interrupt_routine_addr)
- Interrupt status Microcore0 (uc0_irq_status)
- Interrupt status Microcore1 (uc1_irq_status)
- Counters prescaler (counters_prescaler)
- Control register split (control_register_split)

6.15.6.1.1 Flash_enable

Table 120. Flash_enable Registers (0x100, 0x120)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved									checksum_disable	flash_enable	pre_flash_enable	en_dual_uc	dual_uc_failure	chksum_irq_en	chksum_failure
R/W	-									r/w	r	r/w	r/w	r	r/w	r
Lock	-									yes, by pre flash enable	-	yes, by itself	yes, by pre flash enable	-	yes	-
Reset	000000000									0	0	0	0	0	0	0

This 6-bit configuration register includes the following parameters:

- checksum_disable. If set, this bit disables the effects of a failed checksum, so microcore execution is not stopped.
- pre_flash_enable. This bit “freezes” the Code RAM. When this bit is set to ‘1’, the microcontroller cannot further modify the configuration code unless the specific unlock code is written into Unlock_word registers (0x103 and 0x123). It also enables the signature_unit.
- flash_enable. This bit enables the microcores. It can only be set by the signature_unit after a successful checksum calculation.
- en_dual_microcore. This bit is used to enable the dual sequencing mode. Run dual sequencing requires to set the ck_prescaler at least to three (refer to Ck_pre register (0x1C0) section).
- dual_uc_failure. This read-only bit is set to ‘1’ when the dual microcore mode is enabled, but the ck clock is not compatible (ck_prescaler lower than three). The bit is also set if the ck_prescaler value is reduced to a value lower than three while the two microcores on one channel are already running. The bit can only be cleared by unlocking and re-enabling the channel.
- checksum_irq_en. If this bit is ‘1’, the signature unit can generate an interrupt on the IRQ pin of the device in case of Code RAM corruption detected.
- checksum_failure. This read-only bit is set to ‘1’ when a mismatch is found between the calculated checksum and the checksum code stored in the appropriate registers (refer to the Checksum_h registers (0x108, 0x128) and the Checksum_l registers (0x109, 0x129) sections). This bit is set when a checksum calculation fail, even if the checksum is disabled. This bit is reset each time the pre_flash_enable bit is set to ‘1’ to lock the memory.

6.15.6.1.2 Ctrl_reg_uc0

Table 121. Ctrl_reg_uc0 Control Registers for the Microcores 0 (0x101, 0x121)

Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	control_register_shared								control_register							
R/W	configurable r or r/w								r/w							
Lock	no								no							
Reset	00000000								00000000							

- control_register: these eight bits can be used to control the execution of the microprogram of microcore 0, providing control bits that can be read by the microprogram itself. For instance, one bit could be used to enable/ disable recharge pulses on the channel, or to re-enable the actuation after error condition detected.
- control_register_shared: according to a configuration bit stored in the Control_register_split register (0x112, 0x132), these eight bits can be used either as control or like status (similar to the Status_reg_uc0 (0x105, 0x125) registers). In this case, they can only be read through the SPI, while they can be set by the ‘set control register bit’ microcode instruction (stcrb).

6.15.6.1.3 Ctrl_reg_uc1

Table 122. Ctrl_reg_uc1 Control Registers for the Microcores 1(0x102, 0x122)

Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	control_register_shared								control_register							
R/W	configurable r or r/w								r/w							
Lock	no								no							
Reset	00000000								00000000							

- control_register: these eight bits can be used to control the execution of the microprogram of microcore 1, providing control bits that can be read by the microprogram itself. For instance, one bit could be used to enable/ disable recharge pulses on the channel or to re-enable the actuation after an error condition detected.
- control_register_shared: according to a configuration bit stored in the Control_register_split registers (0x112, 0x132), these eight bits can be used either as control or like status (similar to the Status_reg_uc1 (0x106, 0x126) registers). In this case, they can only be read through the SPI, while they can be set by the ‘set control register bit’ microcode instruction (stcrb).

6.15.6.1.4 Unlock_word

The actuation channel execution can be stopped by writing the unlock code at this SPI address. The unlock code is ‘0xBEEF’ (hexadecimal). SPI read operations cannot be performed at this address.

Table 123. Unlock_word Registers (0x103, 0x123)

Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	unlock_word															
R/W	w															
Lock	no															
Reset	-															

6.15.6.1.5 Start_config_reg

This 14-bit configuration register allows for enabling each microcore to the start signals. It is also possible to enable a smart start mode for each microcore (refer to the Start_management Block section for more details).

Table 124. Start_config_reg Registers (0x104, 0x124)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved		smart_ start_ uc1	smart_ start_ uc0	start6_ sens_ uc1	start5_ sens_ uc1	start4_ sens_ uc1	start3_ sens_ uc1	start2_ sens_ uc1	start1_ sens_ uc1	start6_ sens_ uc0	start5_ sens_ uc0	start4_ sens_ uc0	start3_ sens_ uc0	start2_ sens_ uc0	start1_ sens_ uc0
R/W	-		r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	-		yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Reset	00		0	0	0	0	0	0	0	0	0	0	0	0	0	0

- start1_sens_uc0. This bit is ‘1’ if the uc0 is enabled by start1
- start1_sens_uc0. This bit is ‘1’ if the uc0 is enabled by start2
- start3_sens_uc0. This bit is ‘1’ if the uc0 is enabled by start3
- start4_sens_uc0. This bit is ‘1’ if the uc0 is enabled by start4
- start5_sens_uc0. This bit is ‘1’ if the uc0 is enabled by start5
- start6_sens_uc0. This bit is ‘1’ if the uc0 is enabled by start6
- start1_sens_uc1. This bit is ‘1’ if the uc1 is enabled by start1
- start2_sens_uc1. This bit is ‘1’ if the uc1 is enabled by start2
- start3_sens_uc1. This bit is ‘1’ if the uc1 is enabled by start3
- start4_sens_uc1. This bit is ‘1’ if the uc1 is enabled by start4
- start5_sens_uc1. This bit is ‘1’ if the uc1 is enabled by start5
- start6_sens_uc1. This bit is ‘1’ if the uc1 is enabled by start6
- smart_start_uc0. This bit is ‘1’ if the smart start mode is enabled for uc0 (refer to Start_management block section for more details)
- smart_start_uc1. This bit is ‘1’ if the smart start mode is enabled for uc1 (refer to Start_management block section for more details)

6.15.6.1.6 Status_reg_uc0

This 16-bit register is a read-only register and provides information limited to the microcore 0 status to the external microcontroller. The register can be used to exchange application dependent information (status bits, for instance regarding the execution phase of the microprogram) between the microcore and the main microcontroller according to the microprogram definition.

The registers can be configured, such as to be reset after the register SPI read operation (refer to the Reset _behavior register (0x1CE) section).

Table 125. Status_reg_uc0 Registers (0x105, 0x125)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	status_register															
R/W	r															
Lock	-															
Reset on read	configurable															
Reset	0000000000000000															

6.15.6.1.7 Status_reg_uc1

This 16-bit register is a read-only register and provides information limited to the microcore 1 status to the external microcontroller. The register can be used to exchange application dependent information (status bits, for instance regarding the execution phase of the microprogram) between the microcore and the main microcontroller according to the microprogram definition.

The registers can be configured such as to be reset after the register SPI read operation (refer to the Reset_behavior register (0x1CE) section for more details).

Table 126. Status_reg_uc1 Registers (0x106 and 0x126)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	status_register															
R/W	r															
Lock	-															
Reset on read	configurable															
Reset	0000000000000000															

6.15.6.1.8 Code_width

This 10-bit register provides the length of the section of the Code RAM used to store the code. This information has two uses:

- Determination of the length of the special frame transfer used for Code RAM initialization (refer to the Spi_protocol block section for more details). This information is used by the SPI interface.
- Computing the checksum by the signature unit if only a part of the Code RAM is used.

The signature unit only works for code width > 2. Specifying a value in the Code_with register allows the main MCU to partially write the Code RAM.

Table 127. Code_width Registers (0x107, 0x127)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved						code_width									
R/W	-						r/w									
Lock	-						yes									
Reset	000000						0000000000									

6.15.6.1.9 Checksum_h

This 16-bit register contains the 16 MSBs of the checksum of the code contained in the Code RAM. The signature_unit compares the result of its computation to this register and checksum_l.

Table 128. Checksum_h Registers (0x108, 0x128)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	checksum_high															
R/W	r/w															
Lock	yes															
Reset	0000000000000000															

6.15.6.1.10 Checksum_l

This 16-bit register contains the 16 LSBs of the checksum of the code contained in the Code RAM. The signature_unit compares the result of its computation to checksum_h and this register.

Table 129. Checksum_l Registers (0x109, 0x129)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	checksum_low															
R/W	r/w															
Lock	yes															
Reset	0000000000000000															

6.15.6.1.11 Uc0_entry_point

This 10-bit register contains the Code RAM address of the first instruction to be executed by the microcore 0 of the channels 1 and 2.

Table 130. Uc0_entry_point Registers (0x10A, 0x12A)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved						entry_point_address									
R/W	-						r/w									
Lock	-						yes									
Reset	000000						0000001000									

6.15.6.1.12 Uc1_entry_point

This 10-bit register contains the Code RAM address of the first instruction to be executed by the microcore 1 of the channels 1 and 2. This function allows the two microcores to execute completely independent microcodes, while still having the possibility to execute the same program in case the two entry points coincide.

Table 131. Uc1_entry_point Registers (0x10B, 0x12B)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved						entry_point_address									
R/W	-						r/w									
Lock	-						yes									
Reset	000000						0000001000									

6.15.6.1.13 Diag_routine_addr

Table 132. Diag_routine_addr Registers (0x10C, 0x12C)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				diagnosis_routine_address_uc1						diagnosis_routine_address_uc0					
R/W	-				r/w						r/w					
Lock	-				yes						yes					
Reset	0000				000000						000000					

- diagnosis_routine_address_uc0. The complete address is “0000” & “diagnosis routine address uc0”: This is the Code RAM address of the first instruction of the interrupt routine to be executed by uc0 when an automatic diagnosis exception is raised.
- diagnosis_routine_address uc1. The complete address is “0000” & “diagnosis routine address uc1”: This is the Code RAM address of the first instruction of the interrupt routine to be executed by uc1 when an automatic diagnosis exception is raised.

6.15.6.1.14 Driver_disabled_routine_addr

Table 133. Driver_disabled_routine_addr Registers (0x10D, 0x12D)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				driver_disable_routine_address_uc1						driver_disable_routine_address_uc0					
R/W	-				r/w						r/w					
Lock	-				yes						yes					
Reset	0000				000000						000000					

- driver_disable_routine_address_uc0. The complete address is “0000” & “driver disable routine address uc0”: This is the Code RAM address of the first instruction of the interrupt routine to be executed by uc0 when a disabled driver or cksys missing exception is raised.
- driver_disable_routine_address_uc1. The complete address is “0000” & “driver disable routine address uc1”: This is the Code RAM address of the first instruction of the interrupt routine to be executed by uc1 when a disabled driver or cksys missing exception is raised.

The following events can trigger this interrupt (all configurable):

- DRVEN pin is low
- uv_vccp is asserted
- uv_vcc5 is asserted
- uv_vboost is asserted
- cksys is missing
- overtemperature is asserted

6.15.6.1.15 Sw_interrupt_routine_addr

Table 134. Sw_interrupt_routine_addr Registers (0x10E, 0x12E)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	sw_irq_falling_edge_start_uc1	sw_irq_rising_edge_start_uc1	sw_irq_falling_edge start_uc0	sw_irq_rising_edge_start_uc0	software_interrupt_routine_address_uc1						software_interrupt_routine_address_uc0					
R/W	r/w	r/w	r/w	r/w	r/w						r/w					
Lock	yes	yes	yes	yes	yes						yes					
Reset	0	0	0	0	000000						000000					

- software_interrupt_routine_address_uc0: The complete address is “0000” & “software interrupt routine address uc0”: This is the Code RAM address of the first instruction of the interrupt routine to be executed by uc0 when a software interrupt is requested.
- software_interrupt_routine_address_uc1: The complete address is “0000” & “software interrupt routine address uc1”: This is the Code RAM address of the first instruction of the interrupt routine to be executed by uc1 when a software interrupt is requested.
- sw_irq_rising_edge_start_uc0: When this bit is set to ‘1’, the software interrupt 0 is generated towards microcore 0 if a rising edge is detected on the gen_start signal. When set to ‘0’, no software interrupt is required.

- sw_irq_falling_edge start_uc0: When this bit is set to ‘1’, the software interrupt 0 is generated towards microcore 0 if a falling edge is detected on the gen_start signal. When set to ‘0’, no software interrupt is required.
- sw_irq_rising_edge_start_uc1: When this bit is set to ‘1’, the software interrupt 1 is generated towards microcore 1 if a rising edge is detected on the gen_start signal. When set to ‘0’, no software interrupt is required.
- sw_irq_falling_edge_start_uc1: When this bit is set to ‘1’, the software interrupt 1 is generated towards microcore 1 if a falling edge is detected on the gen_start signal. When set to ‘0’, no software interrupt is required.

6.15.6.1.16 Uc0_irq_status

This 13-bit register stores the information about the interrupt currently being served by uc0. If no interrupt is being served, this register is cleared, except for the iret_address field which retains its value until overwritten by the next interrupt.

Table 135. Uc0_irq_status Registers (0x10F, 0x12F)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		interrupt_routine_in_progress	irq_source			iret_address									
R/W	-		r	r			r									
Lock	-		no	no			no									
Reset	00		0	000			0000000000									

- interrupt_routine_in_progress: ‘1’ when an interrupt is being served.
- irq_source:
  - “000”: serving start rising edge interrupt
  - “001”: serving driver disable interrupt request
  - “010”: serving automatic diagnosis interrupt request
  - “011”: serving start falling edge interrupt
  - “100”: serving software interrupt request 0
  - “101”: serving software interrupt request 1
  - “110”: serving software interrupt request 2
  - “111”: serving software interrupt request 3
- iret_address: the value of the return address after the interrupt is served

The return address after an interrupt is always the address where the code execution would have had continued if no interrupt had occurred. For wait and conditional jump instructions, the address is defined taking into account the status of the feedbacks at the moment the interrupt request took place.

6.15.6.1.17 Uc1_irq_status

This 13-bit register stores the information about the interrupt currently being served by uc0. If no interrupt is being served, this register is cleared, except for the iret_address field which retains its value until overwritten by the next interrupt.

Table 136. Uc1_irq_status Registers (0x110, 0x130)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		interrupt_routine_in_progress	irq_source			iret_address									
R/W	-		r	r			r									
Lock	-		no	no			no									
Reset	00		0	000			0000000000									

- interrupt_routine_in_progress: ‘1’ when an interrupt is being served.
- irq_source:
  - “000”: serving start rising edge interrupt

  - “001”: serving driver disable interrupt request
  - “010”: serving automatic diagnosis interrupt request
  - “011”: serving start falling edge interrupt
  - “100”: serving software interrupt request 0
  - “101”: serving software interrupt request 1
  - “110”: serving software interrupt request 2
  - “111”: serving software interrupt request 3
- iret_address: the value of the return address after the interrupt is served

The return address after an interrupt is always the address where the code execution would have had continued if no interrupt had occurred. For wait and conditional jump instructions, the address is defined taking into account the status of the feedbacks at the moment the interrupt request took place.

6.15.6.1.18 Counter_34_prescaler

Table 137. Counter_34_prescaler Registers (0x111, 0x131)

Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	counter_4_per_uc1				counter_3_per_uc1				counter_4_per_uc0				counter_3_per_uc0			
R/W	r/w				r/w				r/w				r/w			
Lock	yes				yes				yes				yes			
Reset	0000				0000				0000				0000			

The counter 3 and 4 of each microcores is base on a multiple of the ck period. The actual ratio is counter_X_per_ucY + 1. For example setting the counter_3_per_uc0 to ‘0100’ results in a time step of counter3 microcore0 of ck period * 5.

6.15.6.1.19 Control_register_split

Table 138. Control_register_split Registers (0x112, 0x132)

Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Reserved														cr_ shared _uc1	cr_ shared _uc0
R/W	-														r/w	r/w
Lock	-														yes	yes
Reset	00000000000000														0	0

- cr_shared_uc0: if set to ‘0’, all the 16 bits of the control register uc0 are used as control bits. If set to ‘1’, the eight MSBs of the control register (control register shared) are used as status bits.
- cr_shared_uc1: if set to ‘0’, all the 16 bits of the control register uc1 are used as control bits. If set to ‘1’, the eight MSBs of the control register (control register shared) are used as status bits.

6.15.6.2 Dual_microcore_arbiter Block

This block handles the access to Code RAM and Data RAM memories by the different possible users:

- the two microcores
- the signature unit (Code RAM only)
- the SPI interface.

6.15.6.2.1 Access Sequence to Code RAM

When the device is operating in single microcore mode, access slots to Code RAM are granted according to Table 139.

Table 139. Code RAM Access Sequence (single microcore mode)

Ck_prescaler	flash_enable	T0	T1	T2	T3	T_even	T_odd
1	1	uc0	CHKSM	-	-	-	-
1	0	SPI r/w	SPI r/w	-	-	-	-
2	1	uc0	CHKSM	-	-	-	-
2	0	SPI r/w	SPI r/w	SPI r/w	-	-	-
3	1	uc0	CHKSM	-	SPI r	-	-
3	0	SPI r/w	SPI r/w	SPI r/w	SPI r/w	-	-
4+	1	uc0	CHKSM	-	SPI r	CHKSM	SPI r
4+	0	SPI r/w	SPI r/w	SPI r/w	SPI r/w	SPI r/w	SPI r/w

The number of access slots is equal to ‘ck_prescaler +1’ (refer to section Ck_per register (0x1C0)). T_even represents all the time slots with an even number id from T4 and following T4. T_odd represents all the time slots with an odd number id from T5 and following T5.

When the device is operating in dual microcore mode, access slots to Code RAM are granted according to Table 140.

Table 140. Code RAM Access Sequence (dual microcore mode)

Ck_prescaler	flash_enable	T0	T1	T2	T3	T_even	T_odd
3	1	uc0	CHKSM	uc1	SPI r	-	-
3	0	SPI r/w	SPI r/w	SPI r/w	SPI r/w	-	-
4+	1	uc0	CHKSM	uc1	SPI r	CHKSM	SPI r
4+	0	SPI r/w	SPI r/w	SPI r/w	SPI r/w	SPI r/w	SPI r/w

Note that dual microcore mode is not operating if ‘ck_prescaler < 3’ (refer to section Ck_per register (0x1C0)).

6.15.6.2.2 Access Sequence to Data RAM

When the device is operating in single microcore mode, access slots to Data RAM are granted according to Table 141.

Table 141. Data RAM Access Sequence (single microcore mode)

Ck_prescaler	flash_enable	T0	T1	T2	T3	T_other	T_last
1	1	SPI r/w	uc0	-	-	-	-
1	0	SPI r/w	SPI r/w	-	-	-	-
2	1	SPI r/w	SPI r/w	uc0	-	-	-
2	0	SPI r/w	SPI r/w	SPI r/w	-	-	-
3	1	SPI r/w	SPI r/w	SPI r/w	uc0	-	-
3	0	SPI r/w	SPI r/w	SPI r/w	SPI r/w	-	-
4+	1	SPI r/w	SPI r/w	SPI r/w	SPI r/w	SPI r/w	uc0
4+	0	SPI r/w	SPI r/w	SPI r/w	SPI r/w	SPI r/w	SPI r/w

The number of access slots is equal to ‘ck_prescaler +1’ (refer to section Ck_per register (0x1C0)). T_last represents the last time slot. T_other represent all time slots (if any) between T3 and T_last.

When the device is operating in dual microcore mode, access slots to Data RAM are granted according to Table 142

Table 142. Data RAM Access Sequence (dual microcore mode)

Ck_prescaler	flash_enable	T0	T1	T2	T3	T_other	T_last
3	1	SPI r/w	uc1	SPI r/w	uc0	-	-
3	0	SPI r/w	SPI r/w	SPI r/w	SPI r/w	-	-
4+	1	SPI r/w	uc1	SPI r/w	SPI r/w	SPI r/w	uc0
4+	0	SPI r/w	SPI r/w	SPI r/w	SPI r/w	SPI r/w	SPI r/w

Note that dual microcore mode is not operating if ‘ck_prescaler < 3’ (refer to section Ck_per register (0x1C0)).

6.15.6.3 Code_RAM Block

The microcode is stored in a 1023x16-bit single port RAM memory called Code RAM. One Code RAM area is dedicated per channel, allowing both uc0 and uc1 to execute this code in parallel if dual microcore mode is enabled.

When enabled, the two microcores can execute either exactly the same code or separate codes, in which case the memory space dedicated to each microcore is a subset of the overall Code RAM. This use of the Code RAM memory is controlled by configuration registers, defining the entry point of each microcore (refer to the Uc0_entry_point (0x10A, 0x12B) and Uc1_entry_point (0x10B, 0x12B) sections).

6.15.6.4 Signature_unit Block

The task of the signature unit is to compute a checksum of the Code RAM to detect possible memory corruption.

The computation is first started when the corresponding Code RAM is locked by the pre_flash_enable bit of the Flash_enable register (0x100 and 0x120). When the computation is complete, the result of the computation is compared to the checksum registers (0x108, 0x109, 0x128, 0x129). These two registers contain the golden checksum provided during the initialization phase through the SPI.

If the result is correct, the signature unit sets the flash_enable bit of the Flash_enable registers (0x0100 and 0x0120), enabling the microcores and a new computation is started again. If the result is not correct, an optional interrupt is issued towards the microcontroller and both microcores accessing the same Code RAM are disabled.

The signature unit can be disabled by writing the appropriate configuration bit in the Flash_enable register. When the signature is disabled, the flash_enable bit is set immediately after the pre_flash_enable bit, and a failed checksum causes a warning (set the appropriate bit in the flash_enable register) without disabling code execution.

The algorithm used for the checksum computation is a pseudo CRC32, according to the standard IEEE 802.3. The polynomial used is x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1 (0x04C11DB7).

The signature unit works only for a code width ≥ 3. Otherwise, the signature unit must be disabled.

In case a checksum calculation failure, the computation can be relaunched by writing again the pre_flash_enable bit to ‘1’.

6.15.6.5 Micro_interface Block

This block named micro_interface contains the Data RAM, a data memory implemented to provide an interface between the external main microcontroller and the internal microcores. This memory is used to provide parameters to the microcores and to return data to the external microcontroller, but it can also be used by the microcores to store temporary data.

This block integrates in particular the following blocks:

- Start_management
- dp_ram, actually instantiating the Data RAM block.

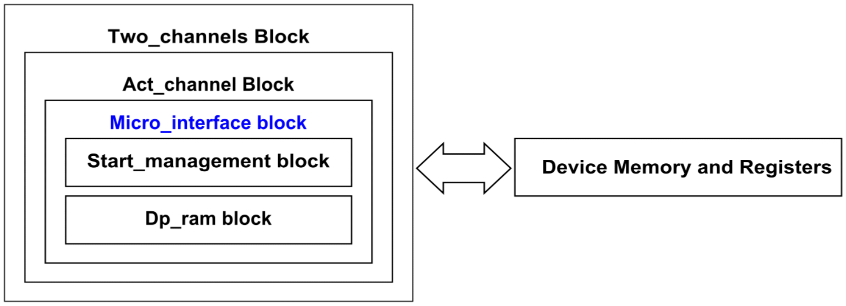


Figure 32. Micro_interface Block Diagram

6.15.6.5.1 Start_management Block

The start_management block is designed to provide an anti-glitch functionality to filter glitches on the input start signal, and also to provide the gen_start_uc0, gen_start_uc1, start_latch_uc0, and start_latch_uc1 signals.

The main purpose of this block is to generate the internal gen_start signals feeding the microcores, starting from the STARTx pins.

Each microcore can be enable by six STARTx pins, according to the enabling map defined the Start_config_reg registers (0x104 and 0x124).

This block also provides the start_latch_ucx signals; these 6-bit signals are used by the corresponding microcore to check which STARTx pin was active when the actuation currently ongoing had begun.

In this way, each microcore can be configured to enabled by all the six STARTx pins, but it also has the possibility to check, while the actuation is ongoing, the level of the STARTx pins in two different modes that can be selected.

The gen_start_ucx and start_latch_ucx can be generated according to two different strategies. The strategies for the two signals can be separately selected in the Start_config_reg registers (0x104 and 0x124):

- Transparent Mode. The gen_start_ucx is high, if at least one of the STARTx signals to which the corresponding microcore is enabled, is high (refer to the Start_config_reg register (0x104 and 0x124) section). The start_latch_ucx signal is a living copy of the six startx pins that can enable the channel.
- Smart Latch Mode. When a STARTx pin (by which the microcore is enable) goes high and the start_latch_ucx is “000000”, the gen_start_ucx is set and the current STARTx pin status is latched in the start_latch_ucx register. If a rising edge is detected on any other STARTx pin, this edge is ignored. The gen_start_ucx signal goes to 0 only when the STARTx pin initially latched goes low. The start_latch_ucx register is reset only by the microcode by means of the rstsl instruction (signal reset_start_latch). The gen_start_ucx signal does not go high, until the start_latch_ucx register has been reset.

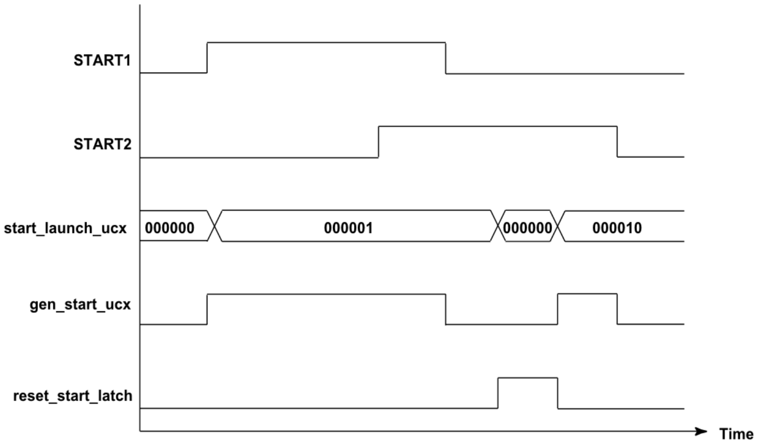


Figure 33. Smart Start Launch Diagram

The gen_start_ucx signals generated by this anti-glitch circuit are then also provided as an input to the corresponding microcores. If Smart Latch mode is enabled, no start edge is latched before the channel is locked by the flash enable bit.

6.15.6.5.2 Dp_ram Block

The data handled by the microcores and by the MCU are stored in the 64x16-bit single port RAM memory called Data RAM. All the 64 Data RAM memory locations can be accessed by the external microcontroller and both microcores of a Logic Channel.

When enabled, the two microcores can access either exactly the same data or separate data, in which case the memory space dedicated to each microcore is a subset of the overall Data RAM. A part of each Data RAM memory can be locked to prevent mishandling by setting the dram1_ private_ area_lock and the dram2_ private_ area_lock bits of the device_lock register (0x1CD).

6.15.6.6 Ch_microcore Block

Each actuation channel block contains two microcores (ch_microcore), a total of four instances in the whole device. Each drives up to six actuators without overlapped actuation, controlling the outputs and acquiring the feedbacks, by means of a microcore structure, conceived to allow full flexibility and programmability of the control strategy.

Each microcore can drive all the device output signals and acquire all the device voltage feedbacks for diagnosis purposes. For safety reasons, the access of each microcore can be restricted to a limited number of output signals. For further details, refer to the Output_switch_box section.

This block integrates mainly the following blocks:

- Instruction_decoder
- Internal_reg_mux
- Counters
- Uprogram_counter
- ALU

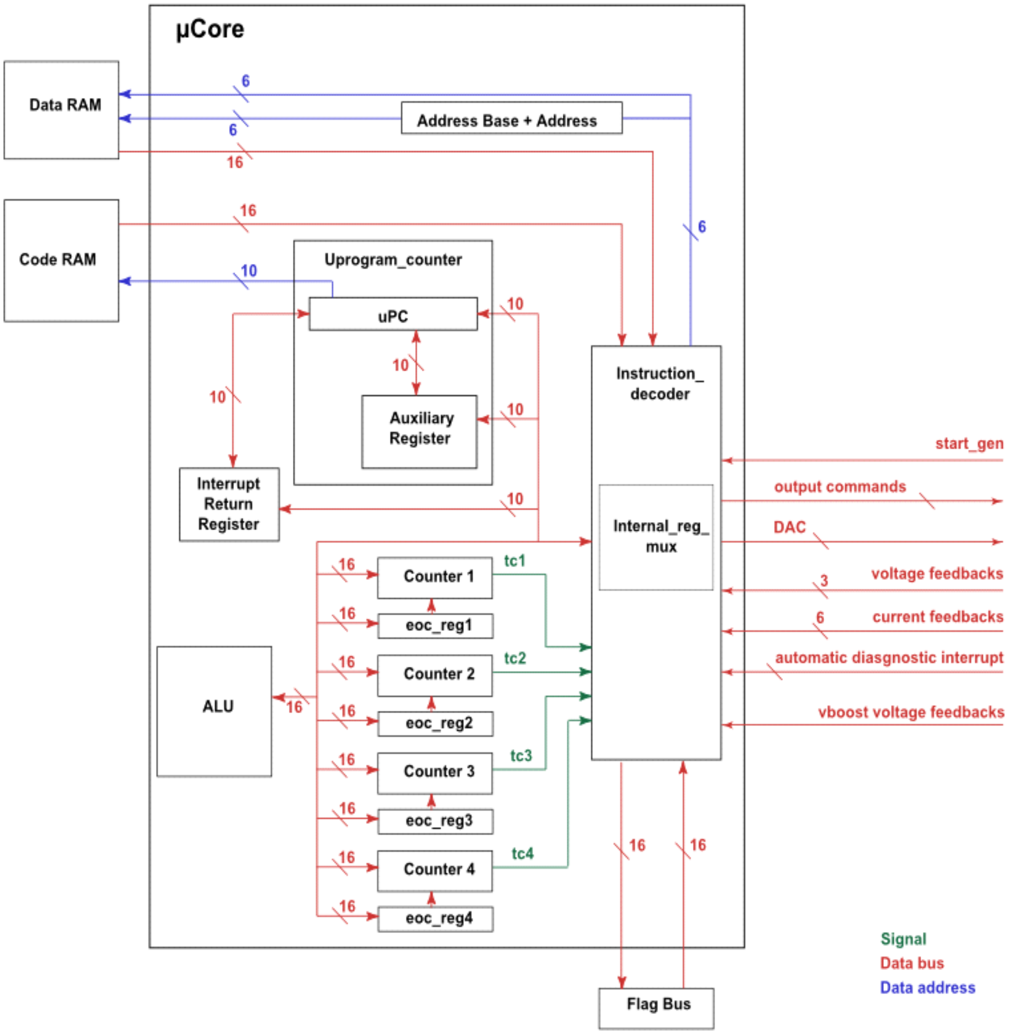


Figure 34. Microcore Block Diagram

6.15.6.6.1 Internal_reg_mux Block

The registers inside the ch_microcore block can be read/written by the Instruction_decoder block through an internal bus. The registers can be accessed by the instructions cp, load, and store. The memory map is shown in Table 143.

Table 143. Microcore Internal Bus Address Map

ADDRESS	NAME	SIZE	DESCRIPTION
0x00	GPR0	16	ALU general purpose register 0 – “r0”
0x01	GPR1	16	ALU general purpose register 1 – “r1”
0x02	GPR2	16	ALU general purpose register 2 – “r2”
0x03	GPR3	16	ALU general purpose register 3 – “r3”
0x04	GPR4	16	ALU general purpose register 4 – “r4”
0x05	GPR5	16	ALU general purpose register 5 – “ir” (immediate register)
0x06	GPR6	16	ALU general purpose register 6 – “mh” (multiplic. result MSBs)
0x07	GPR7	16	ALU general purpose register 7 – “ml” (multiplic. result LSBs)

Table 143. Microcore Internal Bus Address Map (continued)

ADDRESS	NAME	SIZE	DESCRIPTION
0x08	arith_reg	16	ALU condition register
0x09	aux	10	auxiliary register
0x0A	jr1	10	jump register 1
0x0B	jr2	10	jump register 2
0x0C	count1	16	count register of counter 1
0x0D	count2	16	count register of counter 2
0x0E	count3	16	count register of counter 3
0x0F	count4	16	count register of counter 4
0x10	eoc1	16	end of count register of counter 1
0x11	eoc2	16	end of count register of counter 2
0x12	eoc3	16	end of count register of counter 3
0x13	eoc4	16	end of count register of counter 4
0x14	flag	16	flag output of the microcore
0x15	ctrl_reg	16	control register
0x16	status_bits	16	status bits
0x17	spi_data	16	SPI backdoor data register
0x18	dac_sssc	14	dac register same microcore, same channel
0x19	dac_ossc	14	dac register other microcore, same channel
0x1A	dac_ssoc	14	dac register same microcore, other channel
0x1B	dac_osoc	14	dac register other microcore, other channel
0x1C	dac4h4n	12	dac register 4h and 4neg
0x1D	spi_add	8	SPI backdoor address register
0x1E	irq_status	14	interrupt status register
0x1F	ch_rxtx	16	other channel communication register

This is the unique multiplexer (controlled by the Instruction_decoder block output signal) used to select which data write, in case of a transfer from one of the register of the internal bus.

6.15.6.6.2 Counters

These blocks are made of four pairs of a 16-bit up counter and 16-bit end of count registers. Each of the four counters is compared with an eocx (end of count register). If the counter is greater than or equal to its corresponding end of count, then a terminal count signal is asserted. These signals are fed to Instruction_decoder.

Each counter and eocx is set to zero at reset. When a counter reaches its end of count value, counter value incrementation is stopped. If the eocx is changed without resetting the counter value, the counter value continues to increase (if the new end of count value is greater than the counter value) until the new end of count value is reached.

Each of these eight registers is connected to the ch_microcore internal bus (refer to the Internal_reg_mux Block section). These counters can be loaded with data coming from the Data RAM or from the internal bus (e.g. ALU registers). The counters value can be written into the Data RAM or into any the registers connected to the internal bus.

The terminal count register can be updated without stopping the associated counter: This allows on-the-fly data correction in the actuated timings.

All executed load instructions can simultaneously load the eocx with the value specified in the instruction and reset the counter. The counter starts counting up until meeting the eocx value. At this point, an terminal count (tcx) signal is set to inform the microprogram that this event has occurred. The load instructions don’t reset the counter after loading the eocx register.

Counter one and two always operate with the ck execution clock, so the maximum time that is possible to measure is with a single counter is 2^16 * ck clock period (10,923 m at 6.0 MHz).

Counter 3 and 4 can operate with a slower clock, obtained by dividing the execution clock frequency (by an integer factor from 1 to 16), to measure longer times (refer to Conter_34_prescaler section (0x111 and 0x131)). Use these counters results in a lower resolution.

6.15.6.6.3 Uprogram_counter Block

This block instances two registers: the microprogram counter (uPC) and the auxiliary register.

uPC

This is a 10-bit counter used to address the Code RAM containing the microprogram.

After the Code RAM is locked, this counter is loaded with an entry point selected through a SPI register (refer to the Uc0_entry_point (0x10A and 0x12A) and the Uc1_entry_point (0x10B and 0x12B) sections), the address of the first ‘active’ instruction.

If an interrupt is requested, the uPC counter is moved to the appropriate interrupt routine register, as programmed in the parameter registers (refer to Diag_routine_addr (0x10Cand 0x12C) and the Driver_disabled_routine_addr (0x10D and 0x12D) and the Sw_interrupt_routine_addr (0x10E and 0x12E) sections). Only one level of interrupt is supported.

Before entering an interrupt routine, the interrupt status register is latched (refer to the Uc0_irq_status (0x10F and 0x12F) section). When an iret (interrupt return) instruction is executed, the interrupt status register is cleared and the uPC counter can be restored to the original address.

The instruction_decoder block directly controls the uPC in order to allow an efficient management of:

- direct jumps
- conditional jumps
- subroutines execution
- wait states

Auxiliary Register (aux)

This 10-bit register is used to manage the one-level subroutines returns or as an auxiliary memory element.

Any time the system executes a “jump to subroutine” instruction, the uPC is automatically stored in the auxiliary register before jumping to the subroutine start address. When the subroutine execution ends, the incremented auxiliary register content is transferred back to the uPC.

6.15.6.6.4 ALU Block

The microcore contains a simple Arithmetic Logic Unit (ALU). The ALU has an 8-word internal register file, connected to the internal bus. The ALU can perform the following operations:

- Addition and subtraction. These operations are completed in a single ck clock cycle.
- Multiplication. This operation is completed in 17 ck clock cycles. The result is available as a 32-bit number, always in the registers GPR6 (MSBs) and GPR7 (LSBs).
- Shift operations. The operand is shifted of one position (left or right) each ck clock cycle, so it requires from 1 to 16 ck clock cycles to execute. The shift operations always consume the operand. It is also possible to shift an operand by eight positions (left or right) or to swap the eight MSBs with the eight LSBs in one ck clock cycle.
- Logic operation. It is possible to operate a bitwise logical operation (and, or, xor) between an operand and a mask. It is also possible to bitwise invert an operand. All these operations are completed in a single ck clock cycle. These operations always consume the operand.
- C2 conversions. It is possible to convert data from an unsigned representation to two’s complement and vice versa. This operation is completed in a single ck clock cycle.

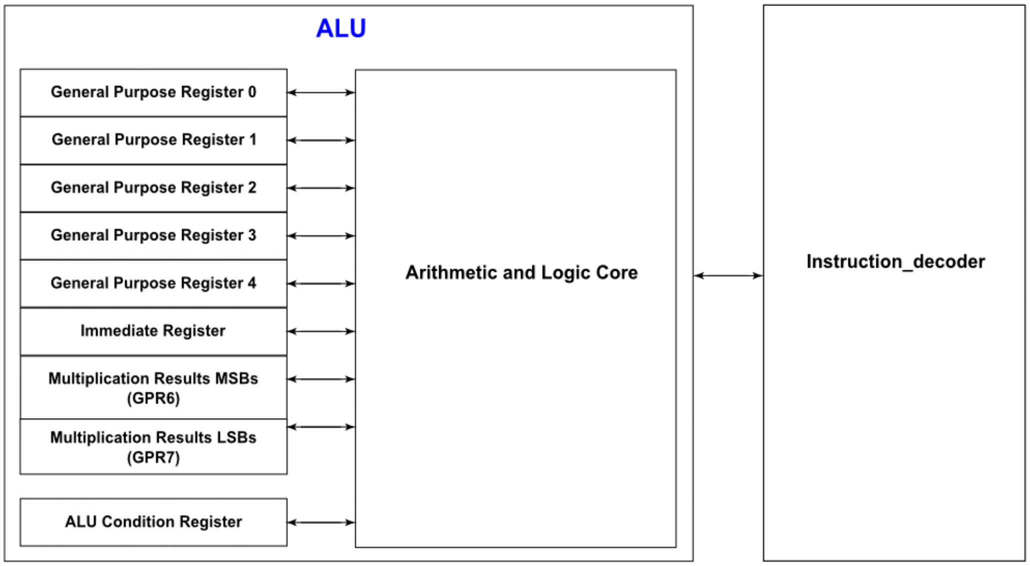


Figure 35. ALU Block Diagram

This operation consumes the operand. While the ALU is busy performing an operation, request of other operations is impossible. In a such cases the request is ignored by the ALU.

The ALU instructions are:

- Addition (add), addition with immediate (addi)
- Subtraction (sub), subtraction with immediate (subi)
- Multiplication (mul), multiplication with immediate (muli)
- Logical operation (and, not, or, xor)
- Conversion from positive to two's complement (toc2) and from two's complement to positive (toint)
- Shift operation (sh32r, sh32l, shl, shr, shls, shrs), shift operation with immediate (sh32ri, sh32li, shli, shri, shlsi, shrsi), and byte manipulation shift (shl8, shr8, swap)
- ALU configuration (stal)

Some ALU instructions are multi-cycle (mul, muli and possibly sh32r, sh32l, shl, shr, shls, shrs, sh32ri, sh32li, shli, shri, shlsi, and shrsi, depending on how many shift positions are required). While a multi-cycle operation is in progress, all ALU instructions are ignored, except for the stal instruction.

During this time, the operations who try to modify the ALU registers (GPR0-7, arith_reg) are ignored (ldirl, ldirh, and possibly cp, load if their destination address is one of the ALU registers). Instructions who try to read the ALU registers are successful (possibly cp and store). Transfer in the ALU register GPR5 constant values present in the microcode is possible by using the ldirl and ldirh instructions.

6.15.6.6.5 Instruction_decoder Block

The Instruction_decoder is in charge of decoding and executing the instructions read from the code memory (Code RAM). The instruction read from the Code RAM is latched by a Instruction_latch register, to reduce the Code RAM time usage to one cksys clock cycle (refer to the Access sequence to Code RAM section for more details).

- This block is enabled by the gen_start signal issue from the start_management block typically used to trigger an actuation
- This block provides the output command, the dac values, the opamp_gain selection to perform the actuation
- Timings is defined through four up-counters whose terminal count (tcx) signal is acquired by the Instruction_decoder block
- This unit can write into the Data RAM data coming from any register connected to the internal bus (refer to the Internal_reg_mux Block section)
- The uPC and the auxiliary register change according to the decoded instruction
- The 16-bit general purpose input-output flag bus is controlled by this unit

- To grant a direct control on the actuation and diagnosis process, this unit can acquire all voltage feedbacks
- In order to allow some control bits exchange, this unit can program a status register, usually used to transfer to the main microcontroller the faults detected on the actuation stage by the diagnosis block
- To acquire a control register, this register allows the external microcontroller to control the microprogram execution flow through microprogram defined control bits
- All the 33816 registers can normally be accessed from the SPI, using a SPI backdoor
- Some instructions can modify the configuration of the microcore or of the device, such as set the end of actuation mode or enable the automatic DCDC mode

All the instructions managed by this block are detailed in the 33816 sections, beginning with CPU Features and Operation.

6.15.6.6.6 Internal Registers Addressing

The Instruction_decoder has access to the internal bus (refer to the Internal_reg_mux Block section) and to a point to point data bus towards the Data RAM. The Instruction_decoder can manage the transfer of data between two internal registers or between one internal register and one element of the Data RAM. The Instruction_decoder cannot directly manage a transfer of data between two elements of the Data RAM.

It is possible to use an offset while addressing the Data RAM. This feature is not available when addressing the internal registers. This offset is contained in the addr_base register and can be added (module 64) to the Data RAM address specified in the instruction. It is possible to modify the value of addr_base with the stab instruction.

The three basic operations are:

- Copy. This instruction copies the value of one of the internal registers to another. The value of addr_base is neglected
- Load. This instruction copies the value of a Data RAM element into one of the internal registers. A boolean parameter specifies if addr_base must be considered while addressing the Data RAM only
- Store. This instruction copies the value of one of the internal registers to a Data RAM element. A boolean parameter specifies if addr_base must be considered while addressing the Data RAM only
- Load instructions can be operated byte wise copying parameters from Data RAM to registers using the following modes:
  - word_read_mode: load instruction transfers 16 bits to an internal register
  - lowbyte_read_mode: load instruction transfers the eight LSBs of the Data RAM value to eight LSBs of an internal register. The eight MSBs of internal register are set to 0x00
  - higbyte_read_mode: load instruction transfers the eight MSBs of the Data RAM value to eight LSBs of an internal register. The eight MSBs of internal register are set to 0x00
  - swapbyte_read_mode: load instruction transfers 16 bits to an internal register but swaps the 8MSBs with LSBs

6.15.6.6.7 Flow Control

This function controls the microprogram counter, selecting the next executed instruction.

Regarding the Flow Control, all events can be classified and described in the following categories:

- Wait instruction. When this instruction is executed, the uPC is frozen (the code execution is stopped) until at least one of the conditions specified in a “wait table” becomes true. Then the uPC is set to the value corresponding to that condition
- Jump instructions. When this instruction is executed, a condition is tested. If the condition is true, then next uPC value is the one specified by the instruction. Otherwise the uPC is incremented by 1
- Other instructions. When all other instructions are executed, the uPC is incremented by 1
- Interrupt requests and returns. When an interrupt request or return is received the uPC is set to a defined location as detailed by the following

This block acquires all the possible conditions checked by the conditional instructions (wait and conditional jump) and checks whether the condition is being matched. Depending on the match of these conditions, the address of the next executed instruction can be:

- uPC: the address doesn’t change. This happens when the instruction being executed is a wait, and none of the enabled conditions were met
- uPC + 1: the address is incremented. This happens for all other instructions, or when the instruction being executed is a conditional jump and the condition was not met

- Jump address: the address is set to the jump address. This happens for all unconditional jump instructions, or for conditional jump instructions when the condition was not met
- Wait destination address: the address is set to the wait destination address (refer to the Wait Instruction section). This happens when the instruction being executed is a wait, and at least one of the enabled conditions has been met; the destination address is the one of the wait entry corresponding to the verified condition. When multiple conditions are satisfied at the same ck clock cycle, the entries with lower id have priority (N°1 is the top priority, N°5 is the lowest priority)
- Automatic diagnosis interrupt routine address: this address (defined in the Diag_routine_addr (0x10C and 0x12C) section) is selected as the new uPC value if an automatic diagnosis interrupt request is received by the microcore. This condition has an higher priority than any instruction and any other interrupt
- Driver disabled interrupt routine address: this address (defined in the Driver_disabled_routine_addr (0x10D and 0x12D) section) is selected as the new uPC value if an interrupt request, due to disabled drivers, is received by the microcore. This condition has an higher priority than any instruction and the software interrupt
- Software interrupt routine address: this address (defined in the Sw_interrupt_routine_addr (0x10E and 0x12E) section) is selected as the new uPC value if a software interrupt request is received by the microcore. This condition has an higher priority than any instruction

Wait Instruction

The wait instruction uses a “wait table” to configure its behavior. The wait table is composed of five entries. Each of the entries contains:

- An enable flag (1-bit). This flag is set by the wait instruction to select if the condition code specified in the entry is enabled
- A condition code. (6-bit) This code specifies the condition that has to be tested
- A destination address (10-bit). This address specifies the address of the Code RAM to which the program execution should jump if the wait condition is met. Regardless of the addressing mode, the address stored in the wait table is always the physical address of the destination

Before the wait instruction is issued, the wait table has to be filled to configure the wait entries to obtain the desired behavior. One instruction is required for each wait entry needing to be configured. The wait table is not reset after each wait, so if some of the entries are common between one wait instruction and the following one, the entry need not to be inserted in the table between the two waits.

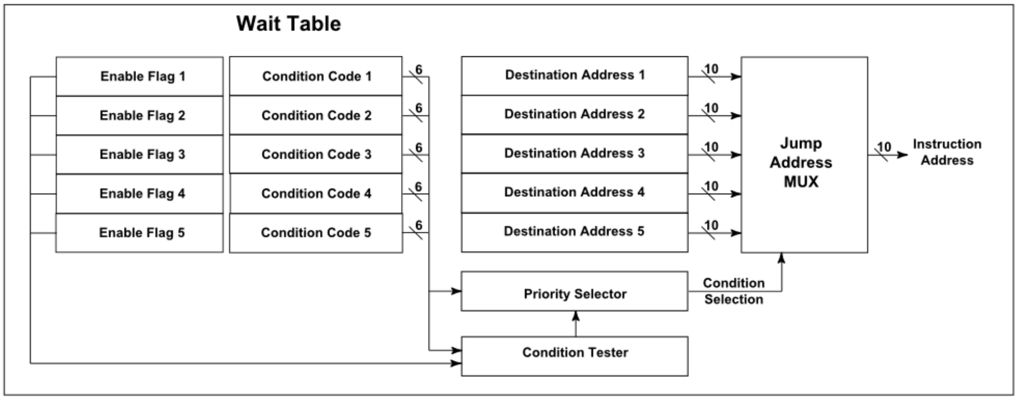


Figure 36. Wait Table Management Diagram

Conditions to be checked by the wait instruction are based on the following inputs:

- terminal_counts: any of the four terminal count (tc1, tc2, tc3, and tc4) signals can be checked to detect if any of the four counters (refer to document Micro Machine Description) has reached its end of count position
- Flags: checks the value (both polarities) of one of the 16 flags signals available
- Shortcut feedback: the voltage feedback (both polarities) related to the three shortcut outputs. For all the three shortcut, it is possible to test the V_{DS} feedback; in addition for all three shortcuts, it is possible to test the Vsource feedback in negative polarity, if available (only if the shortcut is linked to an high-side driver)

- gen_start: checks the value (both polarities) of the filtered chx_start input signal to define when to start and finish an actuation
- current_feedback: the value (both polarities) of the six current feedbacks. Every time a DAC value is changed, the related current feedback is marked as invalid for a fixed time, to avoid using wrong feedback while the DAC is settling. While the feedback is invalid, all the checks related to that signal produce false result, whether the polarity requested by the check itself (jumps are not taken and waits are not quit). For further details refer to current filter registers (0x198, 0x199 and 0x19A) section
- own_current_feedback: the value (both polarities) of the own current feedbacks. This feedback can be different for each microcore and can be changed with the microcode instruction dfcsct. Table 144 shows the configuration after reset. This can be useful when each microcore uses just one (and different) current feedback. This allows exactly the same code in Code RAM, even if each microcore uses a different current measure block

Table 144. Current Feedback Assignment

Microcore	Own current feedback (Reset value)
Uc0Ch1	current feedback 1
Uc1Ch1	current feedback 2
Uc0Ch2	current feedback 3
Uc1Ch2	current feedback 4

- vboost: the output (both polarities) of the comparator that measures the boost voltage. This checks if the boost voltage is above or below the threshold
- op_done: check if a previously issued ALU operation is still in progress or it is completed

Jump Instructions

Conditions to be checked by the jump instructions are the same of the wait instruction with the addition of the following inputs:

- ctrl_reg: checks the value (both polarities) of one of the 16 control bits available in the ctrl_reg register. This is true both when the control register bits are 16 and when eight bits of the control register are used as status register bits (when operating in control_register_split mode the control bits is eight (refer to the Control_register_split (0x112 and 0x132) section)
- status_bits: checks the value (both polarities) of one of the 16 control bits available in the Status_bits register
- voltage feedback: the voltage feedback (both polarities) related to all the outputs
- start_latch: checks the value of the six bit start_latch
- arithmetic_register: checks the value (high polarity only) of one of the bit of the ALU arithmetic register (refer to Arithmetic Condition Register section)
- microcore_id: check if the microcore currently executing is uc0 or uc1

Code RAM Addressing Modes

All the jump instruction have two possible outcomes: if a specific condition (if any) is true, then the code flow continues at a destination specified by a parameter, otherwise it continues to the next code line. In the same way, when a wait entry is configured, a parameter specifies the destination.

The destination is a 10-bit Code RAM address and cannot be directly expressed in the 16-bit instruction, as it is impossible to encode an instruction set with such large parameters.

The instruction set of the 33816 allows only two addressing modes to express the destination parameter for the Code RAM:

- Relative address. The relative address parameter is represented on the five bit only. The physical address of the destination is obtained by adding the relative address to the physical address of the instruction that uses the parameter. The relative address must be considered as 2s-complement, represented and must be extended on 10-bit before the addition

By using relative addresses it is possible to range from “current_address - 16” to “current_address +15”.

- Indirect address. It is possible to jump to the Code RAM address contained into one of two jump_registers (jr1 and jr2): These registers can be loaded with a dedicated instruction and simply referred to in the wait or jump instructions

Interrupt Routines

An interrupt routine is executed when an interrupt request is received by the microcore. The microcore must not already been executing another interrupt routine. The interrupt routine can't be interrupted by any other interrupt, but only be terminated via an iret instruction or (if configured in this way by the iconf instruction) by reading the related diagnosis register through SPI (not through the SPI backdoor):

- Err_ucXchY registers (0x162 to 0x169) for the automatic diagnosis interrupt
- Driver_status register (0x1D2) for the disabled drivers interrupt

The interrupts received are queued while another interrupt execution is on going. When exiting the ongoing interrupt routine with the iret instruction, the queue can be cleared and queued interrupt are ignored. Otherwise, the queued interrupts are executed per their priorities:

- automatic diagnosis interrupt (higher priority)
- driver disabled interrupt
- software interrupt (lower priority)

The interrupt return address is always calculated when the interrupt occurs, and is stored in the Ucx_irq_status registers (0x10F, 0x110, 0x12F, 0x130). The return address is the address where the code execution was interrupted. If a wait or a conditional jump instruction is interrupted, the return address is defined, restoring the status of the feedbacks at the moment the interrupt request occurred.

6.15.6.6.8 HS Feedback Selection

Two of the high-side outputs (HS2 and HS4) have two different V_{DS} feedbacks. One compares the differential voltage between the VBOOST pin and the related high-side source against a threshold, the other one compares the differential voltage between the VBATT pin and the related high-side source against the same threshold.

The two feedbacks can't be used at the same time, but it is possible to selected the desired one by using the slfbk instruction. With the same instruction, it is possible to enable or disable automatic diagnosis on that high-side output. For instance, this can be used to disable automatic diagnosis, when switching to a comparator which is already known to produce an inconsistent feedback.

6.15.6.6.9 DAC Control

The microcore can control four DAC, used to set the values of the thresholds for the current measure blocks. Each DAC is mapped as a register in the internal memory map, so it can be accessed with the load, store, or copy instructions. The internal memory map also contains the DAC_4h4neg register.

Table 145. DAC Register x in DAC Mode

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		dac_offset_x						dac_value_x							
Reset	00		000000						00000000							

Table 146. DAC_4h4neg Register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				dac_value_4neg				dac_value_4h							
Reset	0000				0000				00000000							

While both the dac_value_4neg and dac_value_4h are present in the same register, supplying the value for both the fields every time the register is accessed is not required. By using the instruction stdm parameters:

- If the parameter is dac_access_mode, only the DAC field (the dac_value_4h field for the DAC_4h4neg register) can be read or written
- If the parameter is offset_access_mode, only the offset field (the dac_value_4neg field for the DAC_4h4neg register) can be read or written
- If the parameter is full_access_mode, all the fields can be read or written

The current measure block can perform analog to digital conversion in ADC mode (refer to the ADC conversion registers (0x194, 0x195, 0x196, and 0x197) section). The result of the conversion can be accessed from the internal memory addresses normally used for the DACs (not DAC_4h4neg) if the parameter is set to dac_access_mode, until the ADC mode is disabled.

Table 147. DAC Register x in ADC Mode

Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Reserved								adc_result_x							
Reset	00000000								00000000							

6.15.6.6.10 Microcore Shared Register

It is possible to exchange 16-bit data between different microcores, even belonging to different channels, using the ch_rxtx address in the internal memory map. The transmitting microcore can write its data at this address. The receiving microcore can read the data using the same address, selecting the source microcore with the stcrt instruction.

6.15.6.6.11 Registers Reset

This function generates (when a *rstreg* instruction is executed) a synchronous reset to one of the following registers, according to the parameter that follow the rstreg instruction:

- Reset status bits
- Reset control register
- Reset status bits, automatic diagnosis register and re-enables the possibility to generate automatic diagnosis interrupts
- Reset status bits, control register, automatic diagnosis register and re-enables the possibility to generate automatic diagnosis interrupts
- Reset automatic diagnosis register and re-enables the possibility to generate automatic diagnosis interrupts
- Reset status bits and control register
- Reset status bits and re-enables the possibility to generate automatic diagnosis interrupts
- Re-enables the possibility to generate automatic diagnosis interrupts

6.15.6.6.12 Diagnosis Configuration

According to the parameters of the endiag instruction being executed, this function selects the automatic protection to enable for every output. According to which the output is allowed to be driven by the microcore, the diagnosis enablement or disablement can be restricted.

6.15.6.6.13 Flags

This function sets the value of the 16-bit flag bus coming out of the act_channel block. All the flags controlled by the four microcores are combined according to flags_management setting (refer to the Flags_management Block section). The flag are active low, so the reset value of the flags is ‘1’. When the microcore is unlocked, all the flags are fixed to the inactive state (‘1’) to avoid disturbing the communication of the other microcores.

6.15.6.6.14 Interrupt Request

This function, according to the parameter of the stirq instruction, sets the value of the irq signal managed in the act_channel block and acts on the IRQB pin.

6.15.6.6.15 Subroutine

This function directs the auxiliary register to store the value of the uPC when a jump to subroutine instruction is being executed.

6.15.6.6.16 Current Measure Control

This function:

- Enables and disables the offset compensation of the current measurement analog block when the stoc instruction is being executed (refer to the Offset Compensation (0x190, 0x192, 0x192, and 0x193) section). The offset compensation is performed only on the first four DACs. The offset measured on DAC4l is used also for offset recovery of DAC4h. No offset compensation is foreseen for DAC4neg
- Changes the opamp gain used to measure the voltage across the shunt resistor when the stgn instruction is being executed (refer to the DAC Addressing section)
- Request the current measure block to perform an ADC conversion of the current value by means of the stadc instruction. While in this mode, the current measure block cannot be used to perform threshold measures (refer to the ADC Conversion section)

The microcores are allowed to control the current measurement block using the suitable instructions (refer to the DAC Addressing section).

6.15.6.6.17 Status Bits

This functions sets either the value of the 16-bit status_bits or only one bit in this register when the *stsrb, cp* and *load* instructions are being executed.

6.15.6.6.18 SPI Backdoor

Operation on registers included in the internal memory map and the Data RAM can be performed with most of the instructions. As these locations are often used, access to this location is optimized, in terms of the number of instructions required.

Read and write access to all the registers normally accessible through is possible except the Code RAM and Data RAM by using an SPI backdoor.

To read an SPI register, first the eight LSBs of the address must be provided in the eight LSBs of the ‘SPI address’ at an internal memory map address to the *load* instruction. A read operation must be requested with the *rdspi* instruction. The result is available at the ‘SPI data’ address of the internal memory map.

To write a SPI register, first the eight LSBs of the address must be provided in the eight LSBs of the ‘SPI address’ address, and the data to write must be provided at the ‘SPI data’ address to the *load* instruction. A write operation must be requested with the *wrspi* instruction.

Both the SPI read and write operations are two cycle operations. The registers must not be changed while the operation is in progress.

If the SPI backdoor is not used, the 8-bit register at the address ‘SPI address’ and the 16-bit register at the address ‘SPI data’ can be used as spare register.

SPI Backdoor Access Control

There are some access limitations when requesting write access to SPI registers via the SPI backdoor. It is only possible to write to SPI registers that are not locked at the moment the write operation *wrspi* is requested.

For some special registers, there are additional limitations dependant on the device configuration. Table 148 shows the different limitations. In some cases, the microcore is allowed to change some bits inside a register, but others are not accessible.

Table 148. SPI Backdoor Access Limitations

SPI Registers	Access Rule	Configuration Controlling Access Rule
Vds_threshold_hs, Vsrc_threshold_hs, Vds_threshold_ls_1, Vds_threshold_ls_2	Only microcores which are allowed to control a certain HS or LS pre-driver are allowed to change the corresponding V_{DS} and vsrc threshold. Changes to all other V_{DS} and V_{SRC} values are ignored.	Out_acc_ucXchY
Hs_slewrate, Ls_slewrate	Only microcores which are allowed to control a certain HS or LS pre-driver are allowed to change the corresponding slew rate setting. Changes to all other slew rate settings are ignored.	Out_acc_ucXchY

Table 148. SPI Backdoor Access Limitations

SPI Registers	Access Rule	Configuration Controlling Access Rule
Bias_config	Only microcores which are allowed to control a certain HS or LS pre-driver are allowed to control the corresponding biasing source. Changes to all other biasing sources are ignored.	Out_acc_ucXchY
Current_filter12, Current_filter34l, Current_filter4h4neg, Boost_filter	Only microcores which are allowed to control a certain DAC are allowed to control the corresponding filter setup. Changes to all other filter setups are ignored.	cur_access_1, cur_access_2, boost_dac_access
dac1, dac2, dac3, dac4l, dac4h, dac4neg, boost_dac	No access is possible through the SPI backdoor	-

6.15.6.6.19 Microcore Configuration

DCDC Mode

The DCDC mode (refer to the Low-side Pre-driver for DC-DC Converter (LS7) section) can be enabled from the microcode of any microcore, as long as the microcore has access to the ls7 output.

End Of Actuation

In the final phase of an actuation, while the current in the actuator is decreasing, it is possible to detect when the current has reached the zero value by enabling the actuation mode. In most applications, it is required that the V_{SOURCE} threshold for the corresponding high-side output is set to zero. This condition can be automatically enabled and disabled together with the end of actuation mode.

6.15.7 Input_output_interface Block

Each microcore can access all pins and analog modules of the device. The input signals (e.g. the STARTx pins) and the feedbacks (both from the comparators and from the current measure block) signals are fed to all the microcores. However, an arbitration is needed for the output signals as each microcore provides a complete signal set for all the output resources.

The input_output_interface block combines the four output signal sets as input from the four microcores, and provides a single signal to the analog modules.

This block has also two additional functions:

- The automatic diagnosis, based on the combined output commands and the voltage feedbacks coming from the analog comparators.
- The offset compensation. If requested by the microcores, this block runs the offset compensation algorithm, which uses a small DAC to compensate the input offset of the measurement amplifier.

This block integrates the following blocks:

- boost_dac
- boost_filter
- vds_regfile
- slewrate_regfile
- bias_regfile
- bootstrap_switch_control
- dac_settling_time
- oa_out_config
- dac_switch_box
- output_switch_box

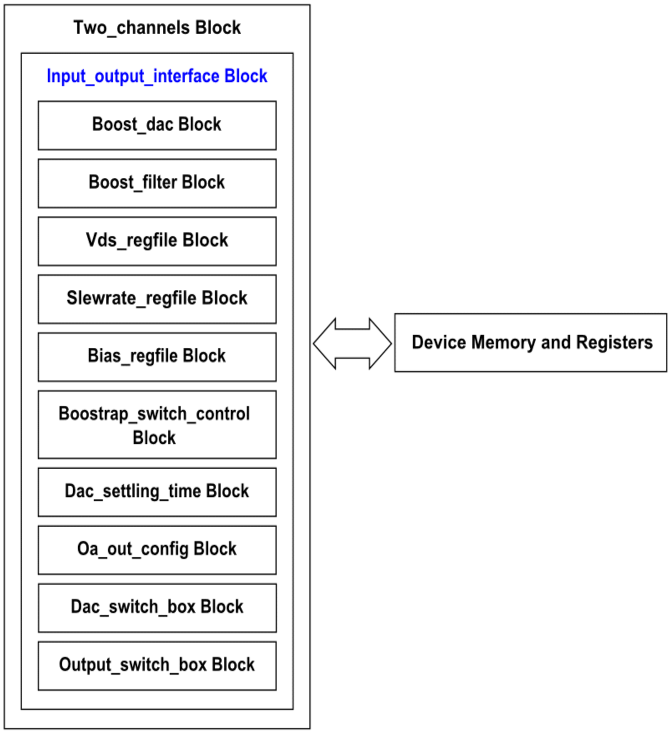


Figure 37. Input_output_interface Block Diagram

6.15.7.1 Boost_dac Block

This block contains the threshold for the boost DAC. This register can be set either from the SPI interface or from a microcores. It is possible to limit the microcore access to the boost_dac register by setting access rights.

End of line offset compensation is provided for the boost monitoring, requiring no microcode operation.

Table 149. Boost_dac Register (0x19B)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								boost_threshold							
R/W	-								r/w							
Lock	-								no							

Table 150. Boost_dac_access Register (0x19C)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved												uc1 ch2 acc	uc0 ch2 acc	uc1 ch1 acc	uc0 ch1 acc
R/W	-												r/w	r/w	r/w	r/w
Lock	-												yes	yes	yes	yes
Reset	000000000000												0	0	0	0

- Boost_threshold. This 8-bit parameter is the threshold used for boost voltage monitoring
- ucX chY acc. This 1-bit parameter (active high) grants access to the dac_boost register

6.15.7.2 Boost_filter Block

This 13-bit register is used to configure the filter for the boost_fbk input signals.

Table 151. Boost_filter Register (0x19D)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved			filter_type	boost_fbk_filter											
R/W	-			r/w	r/w											
Lock	-			yes	yes											
Reset	000			0	000000000000											

- filter_type: This 1-bit parameter selects the type of filter used:
  - if 0 – Any different sample resets the filter counter
  - if 1 – Any different sample decreases the filter counter
- boost_fbk_filter. This 12-bit parameter sets the filtering time for the output of the vboost comparator

The filtering time is: $t_{FTN} = t_{CK}$ x (boost_fbk_filter + 1).

6.15.7.3 Vds_regfile Block

Each comparator threshold is set on three bits. The V_{DS} and V_{SRC} thresholds are defined in Table 42 for the high-side pre-drivers (refer to section 6.5.1, “High-side VDS And VSRC Monitoring") and Table 44 for the low-side pre-drivers (refer to section 6.5.2, “Low-side VDS Monitoring"). These registers can be written through the SPI. The microcores can change the value of each field at runtime, provided that they have the access right to control the related output (refer to the Out_acc_ucX_chY (0x184, 0x185, 0x186, 0x187) section).

Table 152. Vds_threshold_hs Register (0x18A)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	Vds thr Hs5			Vds thr Hs4			Vds thr Hs3			Vds thr Hs2			Vds thr Hs1		
R/W	-	r/w			r/w			r/w			r/w			r/w		
Lock	-	no			no			no			no			no		
Reset	0	000			000			000			000			000		

Table 153. Vsrc_threshold_hs Register (0x18B)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	Vsrc thr Hs5			Vsrc thr Hs4			Vsrc thr Hs3			Vsrc thr Hs2			Vsrc thr Hs1		
R/W	-	r/w			r/w			r/w			r/w			r/w		
Lock	-	no			no			no			no			no		
Reset	0	000			000			000			000			000		

When reading back this register, what is actually read from the SPI is not the content of the register, but the real configuration of the thresholds, in particular the HSx Vsrc thresholds, after the masks imposed by the initialization phase of the bootstrap switch (refer to the Bootstrap_switch_control Block section).

Table 154. Vds_threshold_ls_1 Register (0x18C)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	Vds thr Ls5			Vds thr Ls4			Vds thr Ls3			Vds thr Ls2			Vds thr Ls1		
R/W	-	r/w			r/w			r/w			r/w			r/w		
Lock	-	no			no			no			no			no		
Reset	0	000			000			000			000			000		

Table 155. Vds_threshold_ls_2 Register (0x18D)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved													Vds thr Ls6		
R/W	-													r/w		
Lock	-													no		
Reset	0000000000000													000		

6.15.7.4 Slewrate_regfile Block

These registers store the slew rate configuration value for each output driver. The microcores can change the value of each field at runtime, provided that they have the access right to control the related output (refer to the Out_acc_ucX_chY registers (0x184, 0x185, 0x186, 0x187) section). Each output has the same slew rate for the rising and falling edge (refer to the High-side Pre-drivers Slew Rate Settings and the Low-side Pre-drivers Slew Rate Settings tables) except for the low-side seven (LS7) (refer to the Low-side Seven Pre-drivers PMOS Slew Rate Settings and Low-side Seven Pre-drivers NMOS Slew Rate Settings tables).

Table 156. Hs_slewrate Register (0x18E)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved						slewrate_hs5		slewrate_hs4		slewrate_hs3		slewrate_hs2		slewrate_hs1	
R/W	-						r/w		r/w		r/w		r/w		r/w	
Lock	-						no		no		no		no		no	
Reset	000000						00		00		00		00		00	

Table 157. Ls_slewrate Register (0x18F)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	slewrate_ls7_rising		slewrate_ls7_falling		slewrate_ls6		slewrate_ls5		slewrate_ls4		slewrate_ls3		slewrate_ls2		slewrate_ls1	
R/W	r/w		r/w		r/w		r/w		r/w		r/w		r/w		r/w	
Lock	no		no		no		no		no		no		no		no	
Reset	0		0		0		0		0		0		0		0	

6.15.7.5 Bias_regfile Block

This register configures the biasing for each output which has no biasing except for low-side seven. The microcores can change the value of each field at runtime, considering that they have the access right to control the related output (refer to the Out_acc_ucX_chY registers (0x184, 0x185, 0x186, 0x187).

High-side two and high-side four pre-drivers have two biasing structures, one identical (hsx_en_pu) to the other high-sides and one stronger (hsx_en_s_pu).

Note that when reading back this register, what is actually read from the SPI is not the content of the register, but the real configuration of the high-side and low-side bias, after the masks imposed by the initialization phase of the bootstrap switch (Refer to the Bootstrap_switch_control Block section).

Table 158. Bias_config Register (0x1A4)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved			hs4_en_s_pu	hs2_en_s_pu	ls6_en_pd	ls5_en_pd	ls4_en_pd	ls3_en_pd	ls2_en_pd	ls1_en_pd	hs5_en_pu	hs4_en_pu	hs_3_en_pu	hs2_en_pu	hs1_en_pu
R/W	-			r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	-			no	no	no	no	no	no	no	no	no	no	no	no	no
Reset	000			0	0	0	0	0	0	0	0	0	0	0	0	0

6.15.7.6 Bootstrap_switch_control Block

During initialization phase the control of the bootstrap switch needs to be carefully controlled. In that phase, the following device configurations are affected.

- Hsx_bs_lowcurrent: the low current limit (280 μA), which is set only during initialization independently for each high-side pre-driver
- Vsrc_threshold: the V_{SRC} thresholds of each HSx, which init is set during the first to 0.5 V and after some time to 1.0 V. After the init phase is finished, the V_{SRC} threshold goes back to the value defined in the appropriate register (refer to the Vds_regfile Block section)
- Ls_bias: all the ls_bias are set active for all LSx outputs, during init phase of any high-side pre-driver and then goes back to the configuration defined in the appropriate register (refer to the Bias_regfile section), when all high-side pre-drivers are out of the initialization phase
- Hs_bias: the hs_bias is set inactive for the HSx outputs during initialization and then goes back to the configuration defined in the appropriate register (refer to the Bias_regfile Block section)

During the initialization phase of the bootstrap capacitors, the vccp_external_enable signal is also affected, according to what is defined in the VCCP External Enable Setting Table of the Driver_config register (0x1C5) section. In particular, as long as at least one high-side pre-driver is in bootstrap initialization mode, the vccp_external_enable setting is set to ‘0’ (internal regulator active) if the value of the DBG pin sampled at reset (POResetB and ResetB) was ‘1’.

The charging of the bootstrap capacitors starts after reset is deactivated as soon as the VCCP voltage is ramping up. When the VCCP voltage is above the V_{CCP} undervoltage threshold (uv_vccp=’0’), the state machine of the digital core changes to ‘init_low_thre’ state).

As soon as the VCCP voltage is above the V_{CCP} undervoltage threshold, a global timer is started for all high-side pre-drivers running on cksys with an end of count value of 36 ms. As soon as the timer reaches the end of count value, the Vsrc_threshold is changed from 0.5 to 1.0 V for all the drivers which are still in initialization mode. The fsm for these pre-drivers goes to state ‘init_high_thre’. At the same moment the hsx_src_1V bit is set to ‘1’ for all these drivers.

The bootstrap initialization for each HS pre-driver ends if one of the following conditions is met:

- The bs ready comparator shows that the B_HSx voltage is close to the VCCP voltage, and at the same time the S_HSx voltage is below 0.5 or 1.0 V
- The clamp is activated and at the same time the S_HSx voltage is below 0.5 or 1.0 V
- An LS pre-driver connected to the same high-side pre-driver is switched on (hsx_ls_act signal = ‘1’)
- The connection between low-side pre-drivers and high-side pre-driver is disabled (hsx_ls_act signal = ‘1’)
- The same high-side pre-driver is switched on

Care has to be taken in applications where two high-side pre-drivers are connected to the same node by their S_HSx pin directly or via a diode. It is not allowed in these configurations to turn on the hs_bias via the SPI register or the microcode command before all high-side pre-drivers finished their bootstrap initialization. Otherwise, an active hs_bias from one pre-driver may block the initialization of the other one.

The initialization mode of each high-side pre-driver can be quit by setting the corresponding ‘hsx_ls_act_dis’ bit to ‘1’ (refer to the Hsx_ls_act registers (0x1A6, 0x1A7 and 0x1A8) section). This should be done for each high-side pre-driver not used in an application.

6.15.7.6.1 Bootstrap_charged

This register allows reading the charge status of the high-side bootstrap capacitors during initialization phase.

Table 159. Bootstrap_charged Register (0x1A5)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	bootstrap_init_timer						hs5_ src_ 1V	hs4_ src_ 1V	hs3_ src_ 1V	hs2_ src_ 1V	hs1_ src_ 1V	hs5_ bs_ charged	hs4_ bs_ charged	hs3_ bs_ charged	hs2_ bs_ charged	hs1_ bs_ charged
R/W	r						r	r	r	r	r	r	r	r	r	r
Lock	-						-	-	-	-	-	-	-	-	-	-
Reset	000000						0	0	0	0	0	1	1	1	1	1

- hsx_bs_charged: when ‘0’, the bootstrap capacitor for HSx is charged
- hsx_src_1V: when ‘1’ it was necessary for this pre-driver to switch the V_{SRC} threshold to 1.0 V in order to finish the bootstrap initialization
- bootstrap_init_timer: this shows the current value of the six MSBs of the bootstrap initialization timer. The value is ‘111111’ when the timer is expired

Table 160 details the exact meaning of the bits hsx_bs_charged and hsx_src_1V.

Table 160. Bootstrap_charged Bits

hsx_bs_charged	hsx_src_1V	Description
1	0	Bootstrap capacitor not charged, bootstrap initialization timer not lapsed, V_{SRC} = 0.5 V (reset value)
1	1	Bootstrap capacitor not charged, timer lapsed, V_{SRC} = 1.0 V
0	0	Bootstrap capacitor charged, V_{SRC} = 0.5 V when charging finished
0	1	Bootstrap capacitor charged, V_{SRC} = 1.0 V when charging finished

The bootstrap initialization timer value can be used, together with the other bits of the register, to identify in detail how much time has passed since VCCP voltage was stable, and which threshold is used to detect the charge of the bootstrap capacitor.

Table 161. Bootstrap Initialization Timer

Bit Value	Description
000000	VCCP voltage is not stable (undervoltage)
000001	VCCP voltage is stable since 0.67 ms. Source HS voltage threshold used to detect bootstrap charge is 0.5 V
………	……
100000	VCCP voltage is stable since 21.3 ms. Source HS voltage threshold used to detect bootstrap charge is 0.5 V
………	……
110011	VCCP voltage is stable since 35.3 ms. Source HS voltage threshold used to detect bootstrap charge is 0.5 V
111111	VCCP voltage is stable since at least 36 ms. Source HS voltage threshold used to detect bootstrap charge is 1.0 V

6.15.7.6.2 HS and LS Coupling

Table 162. Hs12_ls_act Register (0x1A6)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	hs2_ ls_act_ dis	hs2_ ls7_act	hs2_ ls6_act	hs2_ ls5_act	hs2_ ls4_act	hs2_ ls3_act	hs2_ ls2_act	hs2_ ls1_act	hs1_ ls_act_ dis	hs1_ ls7_act	hs1_ ls6_act	hs1_ ls5_act	hs1_ ls4_act	hs1_ ls3_act	hs1_ ls2_act	hs1_ ls1_act
R/W	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Reset	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1

This register is used to configure the ground reference of the high-side one and high-side two source pins.

The hs(1/2)_ls_act signal is high if any of the lsx pins connected to the hs(1/2) source pin is switched on or if the function is disabled by the hs(1/2)_ls_act_dis bit.

- hs1_ls1_act: must be set to ‘1’ if ls1 is connected to the same load as hs1
- hs1_ls2_act: must be set to ‘1’ if ls2 is connected to the same load as hs1
- hs1_ls3_act: must be set to ‘1’ if ls3 is connected to the same load as hs1
- hs1_ls4_act: must be set to ‘1’ if ls4 is connected to the same load as hs1
- hs1_ls5_act: must be set to ‘1’ if ls5 is connected to the same load as hs1
- hs1_ls6_act: must be set to ‘1’ if ls6 is connected to the same load as hs1
- hs1_ls7_act: must be set to ‘1’ if ls7 is connected to the same load as hs1
- hs2_ls1_act: must be set to ‘1’ if ls1 is connected to the same load as hs2
- hs2_ls2_act: must be set to ‘1’ if ls2 is connected to the same load as hs2
- hs2_ls3_act: must be set to ‘1’ if ls3 is connected to the same load as hs2
- hs2_ls4_act: must be set to ‘1’ if ls4 is connected to the same load as hs2
- hs2_ls5_act: must be set to ‘1’ if ls5 is connected to the same load as hs2
- hs2_ls6_act: must be set to ‘1’ if ls6 is connected to the same load as hs2
- hs2_ls7_act: must be set to ‘1’ if ls7 is connected to the same load as hs2
- hs1_ls_act_dis: set this bit to disable the link between high-side one and ls predrivers. If this bit is set, the hs1_ls_act signal is forced to ‘1’ regardless if a ls is active
- hs2_ls_act_dis: set this bit to disable the link between high-side two and ls predrivers. If this bit is set, the hs2_ls_act signal is forced to ‘1’ regardless if a ls is active

Table 163. hs34_ls_ act Register (0x1A7)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	hs4_ ls_act_ dis	hs4_ ls7_act	hs4_ ls6_act	hs4_ ls5_act	hs4_ ls4_act	hs4_ ls3_act	hs4_ ls2_act	hs4_ ls1_act	hs3_ ls_act_ dis	hs3_ ls7_act	hs3_ ls6_act	hs3_ ls5_act	hs3_ ls4_act	hs3_ ls3_act	hs3_ ls2_act	hs3_ ls1_act
R/W	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Reset	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1

This register is used to configure the ground reference of the high-side three and high-side four source pins.

The hs(3/4)_ls_act signal is high if any of the lsx pins connected to the hs(3/4) source pin is switched on or if the function is disabled by the hs(3/4)_ls_act_dis bit.

- hs3_ls1_act: must be set to ‘1’ if ls1 is connected to the same load as hs3
- hs3_ls2_act: must be set to ‘1’ if ls2 is connected to the same load as hs3
- hs3_ls3_act: must be set to ‘1’ if ls3 is connected to the same load as hs3
- hs3_ls4_act: must be set to ‘1’ if ls4 is connected to the same load as hs3
- hs3_ls5_act: must be set to ‘1’ if ls5 is connected to the same load as hs3
- hs3_ls6_act: must be set to ‘1’ if ls6 is connected to the same load as hs3

- hs3_ls7_act: must be set to ‘1’ if ls7 is connected to the same load as hs3
- hs4_ls1_act: must be set to ‘1’ if ls1 is connected to the same load as hs4
- hs4_ls2_act: must be set to ‘1’ if ls2 is connected to the same load as hs4
- hs4_ls3_act: must be set to ‘1’ if ls3 is connected to the same load as hs4
- hs4_ls4_act: must be set to ‘1’ if ls4 is connected to the same load as hs4
- hs4_ls5_act: must be set to ‘1’ if ls5 is connected to the same load as hs4
- hs4_ls6_act: must be set to ‘1’ if ls6 is connected to the same load as hs4
- hs4_ls7_act: must be set to ‘1’ if ls7 is connected to the same load as hs4
- hs3_ls_act_dis: set this bit to disable the link between high-side three and ls predrivers. If this bit is set, the hs3_ls_act signal is forced to ‘1’ regardless if a ls is active
- hs4_ls_act_dis: set this bit to disable the link between high-side four and ls predrivers. If this bit is set the hs4_ls_act signal is forced to ‘1’ regardless if a ls is active

Table 164. Hs5_ls_ act Register (0x1A8)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								hs5_ ls_act_ dis	hs5_ ls7_act	hs5_ ls6_act	hs5_ ls5_act	hs5_ ls4_act	hs5_ ls3_act	hs5_ ls2_act	hs5_ ls1_act
R/W	-								r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	-								yes	yes	yes	yes	yes	yes	yes	yes
Reset	00000000								0	1	1	1	1	1	1	1

This register is used to configure the ground reference of the high-side five source pin.

The hs5_ls_act signal is high if any of the lsx pins connected to the high-side five source pin is switched on or if the function is disabled by the hs5_ls_act_dis bit.

- hs5_ls1_act: must be set to ‘1’ if ls1 is connected to the same load as hs5
- hs5_ls2_act: must be set to ‘1’ if ls2 is connected to the same load as hs5
- hs5_ls3_act: must be set to ‘1’ if ls3 is connected to the same load as hs5
- hs5_ls4_act: must be set to ‘1’ if ls4 is connected to the same load as hs5
- hs5_ls5_act: must be set to ‘1’ if ls5 is connected to the same load as hs5
- hs5_ls6_act: must be set to ‘1’ if ls6 is connected to the same load as hs5
- hs5_ls7_act: must be set to ‘1’ if ls7 is connected to the same load as hs5
- hs5_ls_act_dis: set this bit to disable the link between high-side four and ls predrivers. If this bit is set the hs5_ls_act signal is forced to ‘1’ regardless if a ls is active

6.15.7.7 Dac_settling_time Block

This register is used to set the DAC settling time: while this time is being counted no microcode checks on the related current feedback is true as defined in the *wait* Instruction section.

Table 165. Dac_settling_time Register (0x1A9)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										dac_settling_time					
R/W	-										r/w					
Lock	-										yes					
Reset	0000000000										000000					

Every time the value of related DAC register is written, the current feedback is marked as invalid for $t_X = t_{CK}$ x (dac_settling_time + filter_length + 4).

The filter_length value can be set in the current filter registers (0x198, 0x199, and 0x19A).

Due to the fact that the filter configuration (refer to the current filters (0x198, 0x199, and 0x19A) section) can be different for each DAC, also the resulting settling time can be different for each DAC.

6.15.7.8 Oa_out_config Block

These two registers configures the function of the two OA_x pins.

Table 166. Oa_out1_config Register (0x1AA)

Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Reserved										oa1_source			oa1_gain		oa1_en
R/W	-										r/w			r/w		r/w
Lock	-										no			no		no
Reset	0000000000										000			00		0

- oa1_en: when ‘1’ the selected source is sent to the OA_1 pin, otherwise it is put in high-impedance
- oa1_gain: select the gain to apply to the signal.
  - “00”: gain 1.33
  - “01”: gain 2.0
  - “10”: gain 3.0
  - “11”: gain 5.33
- oa1_source: select the signal to send to the OA_1 pin
  - “000”: output from current measurement block 1
  - “001”: output from current measurement block 3
  - “101”: 2.5 Volt

Table 167. Oa_out2_config Register (0x1AB)

Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	Reserved										oa2_source			oa2_gain		oa2_en
R/W	-										r/w			r/w		r/w
Lock	-										no			no		no
Reset	0000000000										000			00		0

- oa2_en: when ‘1’ the selected source is sent to the OA_2 pin, otherwise it is put in high-impedance
- oa2_gain: select the gain to apply to the signal
  - “00”: gain 1.33
  - “01”: gain 2.0
  - “10”: gain 3.0
  - “11”: gain 5.33
- oa2_source: select the signal to send to the OA_2 pin
  - “000”: output from current measurement block 2
  - “001”: output from current measurement block 4
  - “101”: 2.5 Volt

6.15.7.9 Dac_switch_box Block

Each current measure block requires the following control signals:

- DAC value. The 8-bit value identifying the current threshold
- Opamp gain. This 2-bit value identifies the gain of the operational amplifier
- Ofscmp request. This bit identifies if the microcore is requesting to measure the offset
- ADC conversion request. This bit identifies if the microcore is requesting the current measure block to perform an ADC conversion (refer to the ADC Conversion section)

Each microcore produces four of these signals sets, one for each current measure block. This block combines the requests coming from the four microcores in one signal set for each current measure block: the multiple signals sets are managed according access right provided by the cur_access configuration register (refer to the DAC Addressing section for further details).

In addition, the dac_switch_box contains four offset compensation blocks, one for each current measure block. Refer to the Offset Compensation section for further details.

6.15.7.9.1 DAC Addressing

Table 168 shows how the current measurement channels are addressed. This table is fixed and can not be changed.

Table 168. DAC addressing using sssc, ossc, …

Microcore	sssc	ossc	ssoc	osoc
Uc0Ch1	dac1	dac2	dac3	dac4
Uc1Ch1	dac2	dac1	dac4	dac3
Uc0Ch2	dac3	dac4	dac1	dac2
Uc1Ch2	dac4	dac3	dac2	dac1

6.15.7.9.2 Cur_access

This register is designed to provide access rights to manage the control signals (DAC value, opamp gain, ofscomp request) of each current measure block to the required microcores.

Each bit controls the access from one microcore to manage the control signals of a current measure block: if the value is set to one, the microcore can drive those input signals, otherwise access is denied.

Current measure block four is different, as it requires three DAC values instead of just one like the other ones. The acc_ucx_chy_curr_4l bit grants access to all the control signals (DAC value 4L, ofscmp request, opamp gain), except for the DAC values 4H and 4Neg, which are controlled by the acc_ucx_chy_curr_4h_4neg bit.

Table 169. Cur_block_access_1 Register (0x188)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved						acc_uc1_ch1_curr_4h_4neg	acc_uc1_ch1_curr4l	acc_uc1_ch1_curr3	acc_uc1_ch1_curr2	acc_uc1_ch1_curr1	acc_uc0_ch1_curr_4h_4neg	acc_uc0_ch1_curr4l	acc_uc0_ch1_curr3	acc_uc0_ch1_curr2	acc_uc0_ch1_curr1
R/W	-						r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	-						yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Reset	000000						0	0	0	1	0	0	0	0	0	1

Table 170. Cur_block_access_2 Register (0x189)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved						acc_uc1_ch2_curr_4h_4neg	acc_uc1_ch2_curr4l	acc_uc1_ch2_curr3	acc_uc1_ch2_curr2	acc_uc1_ch2_curr1	acc_uc0_ch2_curr_4h_4neg	acc_uc0_ch2_curr4l	acc_uc0_ch2_curr3	acc_uc0_ch2_curr2	acc_uc0_ch2_curr1
R/W	-						r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	-						yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Reset	000000						1	1	0	0	0	0	0	1	0	0

The requests coming from the microcores are not continuous, but they perform a request each time a change to a control signal (DAC value, opamp gain, ofscmp request) is required. In case multiple microcores have access to the same current measure block, as a shared resource, this block is able to handle the collision. If more than one microcore wants to change one of the control signals in the same ck cycle, priorities are used as defined in the Table 171.

If requests to change a control signal are received from different microcores (assuming both have access rights) in different ck cycles, all the requested changes are applied in sequence.

Table 171. Cur_access Collision Handling

Microcore	Priority
Uc0Ch1	1 (highest)
Uc1Ch1	2
Uc0Ch2	3
Uc1Ch2	4 (lowest)

6.15.7.9.3 Current Filters

The six current feedbacks are filtered before feeding them to the microcores. The filters of all the current feedback are independent.

Table 172. Current_filter12 Register (0x198)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				filter_type2	filter_length_2					filter_type1	filter_length_1				
R/W	-				r/w	r/w					r/w	r/w				
Lock	-				yes	yes					yes	yes				
Reset	0000				0	00001					0	00001				

Table 173. Current_filter34l Register (0x199)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				filter_type4L	filter_length_4l					filter_type3	filter_length_3				
R/W	-				r/w	r/w					r/w	r/w				
Lock	-				yes	yes					yes	yes				
Reset	0000				0	00001					0	00001				

Table 174. Current_filter4h4neg Register (0x19A)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				filter_ type4neg	filter_length_4neg					filter_ type4h	filter_length_4h				
R/W	-				r/w	r/w					r/w	r/w				
Lock	-				yes	yes					yes	yes				
Reset	0000				0	00001					0	00001				

- filter_typex. This 1-bit parameter selects the type of filter used for the relative current feedback:
  - if 0 – Any different sample resets the filter counter
  - if 1 – Any different sample decreases the filter counter
- filter_lenght_x. This 5-bit parameter set the filtering time for the current feedback signal

The filtering time is $t_{FTN} = t_{CK}$ x (Filter_lengthx + 1).

6.15.7.9.4 DAC Values

Other than from microcores, it is possible to set the DAC for the current measure blocks by writing these registers.

Table 175. Dac1_value Register (0x19E)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								dac_1_value							
R/W	-								r/w							
Lock	-								no							
Reset	00000000								00000000							

Table 176. Dac2_value Register (0x19F)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								dac_2_value							
R/W	-								r/w							
Lock	-								no							
Reset	00000000								00000000							

Table 177. Dac3_value Register (0x1A0)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								dac_3_value							
R/W	-								r/w							
Lock	-								no							
Reset	00000000								00000000							

Table 178. Dac4l_value Register (0x1A1)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								dac_4l_value							
R/W	-								r/w							
Lock	-								no							
Reset	00000000								00000000							

Table 179. Dac4h_value Register (0x1A2)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								dac_4h_value							
R/W	-								r/w							
Lock	-								no							
Reset	00000000								00000000							

Table 180. Dac4neg_value Register (0x1A3)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								dac_4neg_value							
R/W	-								r/w							
Lock	-								no							
Reset	00000000								00000000							

6.15.7.9.5 ADC Conversion

Each current measure block can perform ADC conversion (refer to Figure 20). A conversion is performed when requested by a microcore (refer to the Current Measure Control section) with the correct access rights (refer to the DAC Addressing section).

The DAC4L is used when performing an ADC conversion using current measurement channel four.

For the ADC mode, a signal path via the OAx multiplexer, a track and hold circuit, the OAx amplifier, and the DAC feedback multiplexer are used driven by the adc_modex signal. Therefore while using ADC mode on current measurement channel one and three the OA1 output is blocked, while using ADC mode on channel two or four, the OA2 output is blocked. The OAx multiplexer has to be set to the right input and the OAx output has to be enabled manually. The OAx amplifier is set to a gain of 1.0 automatically. It is not possible to do ADC conversion at the same time at channel one and three, or on channel two and four.

The conversion takes 11 ck_ofscmp clock cycles (refer to the Clock Offset Compensation Prescaler section). Four ck_ofscmp clock cycles are needed for the first bit, because the OAx amplifier output has to settle first after changing the gain. After the first bit, one clock cycle is needed for every of the seven following bits. The 33816 has a ‘track and hold’ circuit for the ADC mode. The switch of the track and hold circuit is opened before the ADC conversion starts and is closed again when ADC mode is switched off. The result of the conversion is stored in the corresponding adc register after the conversion is finished. It is available to the microcore as long as the ADC mode is on and available via the SPI register until the next ADC conversion is started.

To trigger a new conversion, the ADC mode must be switched off and on again.

The result can be read via the SPI registers from the external microcontroller, or from every microcore via the internal register map (refer to the DAC Control section).

Table 181. Adc1_result Register (0x194)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								conversion_1_value							
R/W	-								r/w							
Lock	-								no							
Reset	00000000								10000000							

Table 182. Adc2_result Register (0x195)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								conversion_2_value							
R/W	-								r/w							
Lock	-								no							
Reset	00000000								10000000							

Table 183. Adc3_result Register (0x196)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								conversion_3_value							
R/W	-								r/w							
Lock	-								no							
Reset	00000000								10000000							

Table 184. Adc4_result Register (0x197)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								conversion_4_value							
R/W	-								r/w							
Lock	-								no							
Reset	00000000								10000000							

6.15.7.9.6 Offset Compensation

The offset can be measured on each current measure block, including opamp, DAC, and comparator. For current measure block four, only comparator 4l, the relative opamp and the DAC 4l are considered. The measured offset is automatically compensated during normal operation. The compensation must be enabled by the microcores by means of the *stoc* instruction (when the input current to the current measurement block is null) through the combined ofs_comp signal.

At the end of the measurement sequence, a new offset register value is stored until the next time this measurement sequence is executed.

The procedure can be interrupted at any time. Even a partial run of the offset compensation procedure is guaranteed to produce an offset measure that is never worse than the precedent one.

The current measurement channel DAC value is set automatically to a value of 0x1A which corresponds to an output voltage of 253.9 mV to perform the offset compensation. For current measurement channel four, the DAC 4l and feedback 4l are used for the offset compensation.

The offset can be read through the SPI registers. It is also possible to change the value compensated by writing to these registers. If the offset compensation is requested by microcores, it starts from the precedent result (or from the data forced through the SPI). As the offset can be both positive and negative, all the values in these registers are represented as two's complement.

Table 185. Offset_compensation1 Register (0x190)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										offset_current_measure_block_1					
R/W	-										r/w					
Lock	-										no					
Reset	000000000										000000					

Table 186. Offset_compensation2 Register (0x191)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										offset_current_measure_block_2					
R/W	-										r/w					
Lock	-										no					
Reset	000000000										000000					

Table 187. Offset_compensation3 Register (0x192)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										offset_current_measure_block_3					
R/W	-										r/w					
Lock	-										no					
Reset	000000000										000000					

Table 188. Offset_compensation4 Register (0x193)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										offset_current_measure_block_4					
R/W	-										r/w					
Lock	-										no					
Reset	000000000										000000					

The offset values are stored in registers using the twos complement notation. The values can range from -32 to 31. The value is then converted to sign-module notation before being transferred to the analog section.

6.15.7.10 Output_switch_box Block

The 33816 can drive two types of outputs: high-side and low-side. Each type of output requires different control signals.

The low-side outputs require the following control signals:

- Output value. The value to be driven on the gate of the external MOSFET
- V_{DS} threshold. A 3-bit signal that selects the threshold for the comparator that measures the drain-source voltage of the external MOSFET
- Automatic freewheeling (for low-side pre-drivers 4, 5, 6 and 7). Four low-sides pre-drivers can be configured to work as automatic freewheeling drivers
- En_halt_vds. This signal enables (if set to 1) the automatic coherency check between the output and the V_{DS} comparator
- The high-side outputs require the following control signals:
- Output value. The value to be driven on the gate of the external MOSFET

- V_{DS} threshold. A 3-bit signal that selects the threshold for the comparator that measures the drain-source voltage of the external MOSFET
- V_{SRC} threshold. A 3-bit signal that selects the threshold for the comparator that measures the source-ground voltage of the external MOSFET
- Automatic freewheeling (for high-side pre-driver 5 only). One high-side pre-driver can be configured to work as an automatic freewheeling driver
- En_halt_vds. This signal enables (if set to 1) the automatic coherency check between the output and the V_{DS} comparator
- En_halt_src. This signal enables (if set to 1) the automatic coherency check between the output and the V_{SRC} comparator

Each microcore produces seven signals sets for the low-sides and five signals sets for the high-sides.

This block combines the requests coming from the four microcores in one signal set for each output: the multiple signals sets are managed according access right provided by the four Out_acc_ucXchY configuration registers.

This block integrates the following blocks:

- output_access
- dcdc_convert_control
- automatic_diagnosis
- output_routing
- error_handler

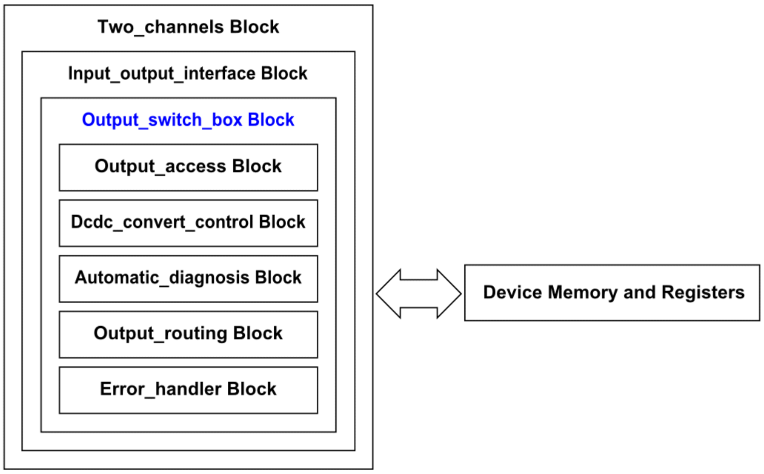


Figure 38. Output_switch_box Block Diagram

6.15.7.10.1 Output_access Block

This block configured by the four out_acc_seqX_chY registers is designed to provide access rights to manage the control signals (output_command, Vds threshold, Vsrc threshold, automatic freewheeling, en_halt_x) of each output block to the required microcores.

Each bit controls the access from one microcore to manage the control signals of an output: if the value is set to ‘1’, the microcore can drive the control signals (output_command, Vds threshold, Vsrc threshold, automatic freewheeling, en_halt_x), otherwise access is denied.

Table 189. Out_acc_uc0_ch1 Register (0x184)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				acc_uc0_ch1_ls7	acc_uc0_ch1_ls6	acc_uc0_ch1_ls5	acc_uc0_ch1_ls4	acc_uc0_ch1_ls3	acc_uc0_ch1_ls2	acc_uc0_ch1_ls1	acc_|uc0_ch1_hs5	acc_uc0_ch1_hs4	acc_uc0_ch1_hs3	acc_uc0_ch1_hs2	acc_uc0_ch1_hs1
R/W	-				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	-				yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Reset	0000				0	0	1	0	0	0	1	0	0	0	0	1

Table 190. Out_acc_uc1_ch1 Register (0x185)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				acc_uc1_ch1_ls7	acc_uc1_ch1_ls6	acc_uc1_ch1_ls5	acc_uc1_ch1_ls4	acc_uc1_ch1_ls3	acc_uc1_ch1_ls2	acc_uc1_ch1_ls1	acc_uc1_ch1_hs5	acc_uc1_ch1_hs4	acc_uc1_ch1_hs3	acc_uc1_ch1_hs2	acc_uc1_ch1_hs1
R/W	-				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	-				yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Reset	0000				0	0	1	0	0	0	1	0	0	0	0	1

Table 191. Out_acc_uc0_ch2 Register (0x186)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				acc_uc0_ch2_ls7	acc_uc0_ch2_ls6	acc_uc0_ch2_ls5	acc_uc0_ch2_ls4	acc_uc0_ch2_ls3	acc_uc0_ch2_ls2	acc_uc0_ch2_ls1	acc_uc0_ch2_hs5	acc_uc0_ch2_hs4	acc_uc0_ch2_hs3	acc_uc0_ch2_hs2	acc_uc0_ch2_hs1
R/W	-				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	-				yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Reset	0000				0	0	1	0	0	0	1	0	0	0	0	1

Table 192. Out_acc_uc1_ch2 Register (0x187)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				acc_uc1_ch2_ls7	acc_uc1_ch2_ls6	acc_uc1_ch2_ls5	acc_uc1_ch2_ls4	acc_uc1_ch2_ls3	acc_uc1_ch2_ls2	acc_uc1_ch2_ls1	acc_uc1_ch2_hs5	acc_uc1_ch2_hs4	acc_uc1_ch2_hs3	acc_uc1_ch2_hs2	acc_uc1_ch2_hs1
R/W	-				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	-				yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Reset	0000				0	0	1	0	0	0	1	0	0	0	0	1

The requests coming from the microcores are not continuous. Instead the microcores perform a request each time a change to a control signal (output_command, V_{DS} threshold, V_{SRC} threshold, automatic freewheeling, en_halt_x) is required. In case multiple microcores have access to the same output block, as a shared resource, this block is able to handle the collision: if more than one microcore wants to change one of the control signals in the same ck cycle, priorities are used as defined in Table 193.

If one of the microcores which has access to a pre-driver is not locked, the other microcore can switch on the pre-driver only for one ck cycle maximum. After one ck cycle the output is switched off again, because this request comes from the disabled microcore. This is a safety feature of the device.

If requests to change a control signal are received from different microcores (assuming both have access rights) in different ck cycles, all the requested changes are applied in sequence.

Table 193. Out_acc_ucXchY Collision Handling

Microcore	Priority
Uc0Ch1	1 (highest)
Uc1Ch1	2
Uc0Ch2	3
Uc1Ch2	4 (lowest)

6.15.7.10.2 Dcdc_conver_control Block

This mode can be enabled by a microcode instruction, and can be used to achieve a very fast current regulation between the current thresholds 4H (higher limit) and 4L (lower limit).These two current thresholds can be supplied either from microcode or by writing the DAC register (refer to the DAC Values section).

LS7 output is switched on when current_feedback_4L is low and it is switched off when current_feedback_4H is high. The path from the shunt resistor to the LS7 output is completely asynchronous to any clock (ck and cksys) of the device.

The current feedback of DAC 4H takes priority, so in the case both feedbacks are active (DAC 4H feedback high and DAC 4L feedback is low), the output LS7 is driven low.

The operation of this automatic DC/DC converter control can be controlled by the microcode instruction *stdcctl*. Every microcore that has access to the LS7 pre-driver. according to the crossbar configuration, can switch the automatic DC/DC control on or off by using this microcode instruction. The LS7 output can be controlled by the automatic DC/DC converter control or by the standard control method (microcode instructions).

As soon as a microcore which has access to the LS7 is unlocked, the automatic DC/DC control is switched off.

6.15.7.10.3 Automatic_diagnosis Block

This block named automatic_diagnosis performs a coherency check between an output and the related V_{DS} feedback (for all the outputs) and V_{SRC} feedback (for the high-side outputs only).

The error_feed produced by each automatic_diagnosis block is fed to the error_handler block (refer to the Error_handler Block section).

The Automatic_diagnosis block integrates the following blocks:

- Filter_input
- Channel_check
- Error_handler

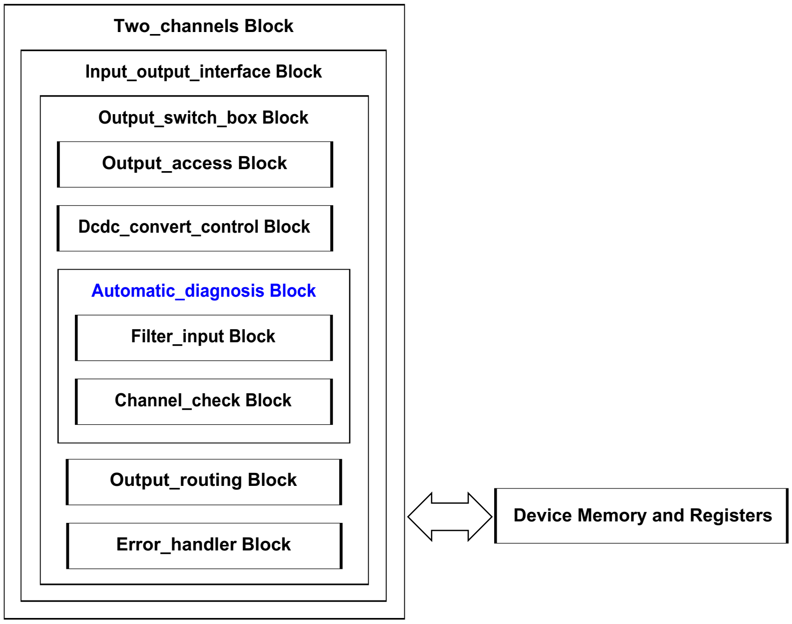


Figure 39. Automatic_diagnosis Block Diagram

6.15.7.10.4 Automatic Diagnosis Reaction Time

If the disable window is exceeded, and the automatic diagnosis detects an error between the HSx_in and the filtered HSx_Vds_fbk signal, an interrupt is generated towards the microcore. Due to this interrupt the program counter of the microcore is set to the first instruction of the error routine.

It takes four ck cycles (666 ns at 6.0 MHz) until the execution of the first microcode operation of the error routine is completed.

It means that if the first microcode command is used to switch off all pre-drivers, this action is delayed by four ck cycles. In more detail, it takes one ck cycle to detect the error, one ck cycle to generate the interrupt, one ck cycle to move the program counter to the error routine and one ck cycle to execute the first instruction.

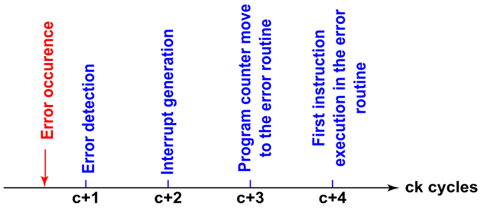


Figure 40. Diagnosis Reaction Time Diagram

6.15.7.10.5 LSx Output Registers

These registers define the automatic diagnosis parameter and output routing option from the low-side X output.

Table 194. Lsx_diag_config1 Registers (0x140, 0x143, 0x146, 0x149, 0x14C, 0x14F)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		filter_type	filter_length						disable_window						
R/W	-		r/w	r/w						r/w						
Lock	-		yes	yes						yes						
Reset	00		0	000000						0000000						

Table 195. Lsx_diag_config2 Registers (0x141, 0x144, 0x147, 0x14A, 0x14D, 0x150)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved												error_table			
R/W	-												r/w			
Lock	-												yes			
Reset	000000000000												0000			

Table 196. Lsx_output_config Registers (0x142, 0x145, 0x148, 0x14B, 0x14E, 0x151)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved											output_routing				inv
R/W	-											r/w				r/w
Lock	-											yes				yes
Reset	00000000000											1111				0

- filter_type. This 1-bit parameter selects the type of filter used:
  - If 0 – Any different sample resets the filter counter
  - If 1 – Any different sample decreases the filter counter
- filter_lenght: this 6-bit parameter set the filtering time for the input feedback signal. The filtering time is: $t_{FTN} = t_{CK}$ x (filter_length + 1)
- error_table: this 4-bit parameter defines the logical value of an error signal, issued from the output and the related V_{DS} feedback signal. Basically, this table defines the output of the coherency check between the driven output and the acquired feedback; a logic one value means there is no coherency in the check, and then an error signal towards the microcore should be generated

Table 197. LSx V_{DS} Error Table Selection True Table

	output_command = 0 (Pre-driver switched OFF)	**output_command = 1 (Pre-driver switched ON)**
lsx_vds_fbk = 0 (V_{DS} below threshold)	error_table (0)	error_table (2)
lsx_vds_fbk = 1 (V_{DS} above threshold)	error_table (1)	error_table (3)

- disable_window: this 7-bit parameter configures a time period during which any check on the LSx_Vds_feed signal is disabled after any change on the output_command signal. $t_{DTL} = t_{CK}$ x (Disable_window + 4)
- output_routing: this 4-bit parameter defines if the LSx output is controlled by the microcores or by an input flag pin. This function is not active if ck_per = 0.

Table 198. LSx Output Control Table

output_routing	output_command
0	driven from flag0
1	driven from flag1
2	driven from flag2
3	driven from flag3
4	driven from flag4
5	driven from flag5
6	driven from flag6
7	driven from flag7
8	driven from flag8
9	driven from flag9
10	driven from flag10
11	driven from flag11
12	driven from flag12
13	driven from the microcores
14	
15	

- inv: this parameter inverts the polarity of the LSx output signal, with respect to the polarity defined by the microcore. This affects the output command towards the pre-drivers, but the error_table of the associated feedback is not affected since diagnosis already takes into account the pre-driver status (even when the invert bit is set). This function is not available in case of direct gate drive by input flag pin.

6.15.7.10.6 LS7 Output Register

Table 199. Ls7_output_config Register (0x152)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										fast_ dcdc_ en	output_routing				inv
R/W	-										r	r/w				r/w
Lock	-										-	yes				yes
Reset	0000000000										0	1111				0

- output_routing. This four bit parameter defines if the LSx output is controlled by the microcores or by an input flag pin. This function is not active if ck_per = 0.

Table 200. LSx Output Control Table

output_routing	output_command
0	driven from flag0
1	driven from flag1
2	driven from flag2
3	driven from flag3
4	driven from flag4

Table 200. LSx Output Control Table

output_routing	output_command
5	driven from flag5
6	driven from flag6
7	driven from flag7
8	driven from flag8
9	driven from flag9
10	driven from flag10
11	driven from flag11
12	driven from flag12
13	driven from the microcores
14	
15	

- Invert. This parameter inverts the polarity of the LSx output signal, with respect to the polarity defined by the microcore. This function is not available in case of direct gate drive by input flag pin.
- fast_dcdc_en. This bit is set when the automatic DC-DC control feature for LS7 is enabled (Refer to the Dcdc_conver_control Block section for the behavior of LS7 during this mode)

6.15.7.10.7 HSx Output Register

These registers define the automatic diagnosis parameter and output routing option for the high-side X output.

Table 201. Hsx_diag_config_1 Registers (0x153, 0x156, 0x159, 0x15C, 0x15F)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		Filter_type	filter_length						disable_window						
R/W	-		r/w	r/w						r/w						
Lock	-		yes	yes						yes						
Reset	00		0	000000						0000000						

Table 202. Hsx_diag_config_2 Registers (0x154, 0x157, 0x115A, 0x15D, 0x160)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								error_table_src				error_table_vds			
R/W	-								r/w				r/w			
Lock	-								yes				yes			
Reset	00000000								0000				0000			

Table 203. Hsx_output_config Registers (0x155, 0x158, 0x15B, 0x15E, 0x161)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved							dead_time				output_routing				inv
R/W	-							r/w				r/w				r/w
Lock	-							yes				yes				yes
Reset	000000000							0000				1111				0

- error_table_vds: This 4-bit parameter defines the logical value of an error signal, starting from the output and the related V_{DS} feedback signal. Basically, this table defines the output of the coherency check between the driven output and the acquired feedback; a 'logic one value' means there is no coherency in the check and then an error signal towards the microcore should be generated

Table 204. HSx V_{DS} Error Table Selection True Table

	output_command = 0 **(Pre-driver switched OFF)**	**output_command = 1** **(Pre-driver switched ON)**
hsx_vds_fbk = 0 (Vds below threshold)	error_table_vds (0)	error_table_vds (2)
hsx_vds_fbk= 1 (Vds above threshold)	error_table_vds (1)	error_table_vds (3)

- disable_window: This 7-bit parameter configures a time period during which any check on the HSx_Vds_feed and HSx_Vsrc_feed signals is disabled after any change on the output_command signal. $t_{DTL} = t_{CK}$ x (Disable_window + 4)
- error_table_src: This 4-bit parameter defines the logical value of an error signal, starting from the output, and the related V_{SRC} feedback signal. Basically this table defines the output of the coherency check between the driven output and the acquired feedback; a logic 1 value means there is no coherency in the check and then an error signal towards the microcore should be generated

Table 205. HSx Vsrc Error Table Selection True Table

	output_command = 0 **(Pre-driver switched OFF)**	**output_command = 1** **(Pre-driver switched ON)**
hsx_src_fbk = 0 (Vsrc below threshold)	error_table_src (0)	error_table_src (2)
hsx_src_fbk = 1 (Vsrc above threshold)	error_table_src (1)	error_table_src (3)

- filter_type: This 1-bit parameter selects the type of filter used: if 0 – Any different sample resets the filter counter if '1' – Any different sample decreases the filter counter
- dead_time: This 5-bit register is used to store the value of the dead_time end of count used in the generation of the free wheeling output (delay between the high-side output and the free wheeling output). The freewheeling command goes high after a programmable time (t_{FWDLY}) with respect to the high-side falling edge. In this mode, the high-side command rising edge is always delayed of the same programmable time (t_{FWDLY}) with respect to the rising edge requested by the microcores
  $t_{FWDLY} = t_{CK}$ x (Dead_time + 1)
- output_routing: This 4-bit parameter defines if the HSx output is controlled by the microcores or by an input flag pin. This function is not active if ck_per = 0.

Table 206. HSx Output Control Table

output_routing	**output_command**
0	driven from flag0
1	driven from flag1
2	driven from flag2
3	driven from flag3
4	driven from flag4
5	driven from flag5
6	driven from flag6
7	driven from flag7
8	driven from flag8
9	driven from flag9
10	driven from flag10
11	driven from flag11

Table 206. HSx Output Control Table (continued)

output_routing	output_command
12	driven from flag12
13	driven from the microcores
14	
15	

- inv: This parameter inverts the polarity of the HSx output signal, with respect to the polarity defined by the microcore. This affects the output command towards the pre-drivers, but the error_table of the associated feedback is not affected since diagnosis already takes into account the pre-driver status (even when the invert bit is set). This function is not available in case of direct gate drive by input flag pin.
- filter_lenght: This 6-bit parameter set the filtering time for the input feedback signal. The filtering time is: t_{FTN} = t_{CK} x (Filter_length + 1)

6.15.7.10.8 Filter_input Block

This block is provided to filter the input feedback coming from each analog comparator.

The input filtering scheme adopted is based on a 6-bit counter clocked by the ck signal (refer to the Clock Prescaler section). After a new level is first detected on the input signal, at least N consecutive samples must be at that level before it is recognized as a valid transition. This filter operates the same way on the rising and the falling edge of the input signal.

N is a programmable 6-bit value provided by signal filter_length, (refer to the LSx Output Registers and the HSx Output Registers sections).

6.15.7.10.9 Channel_check Block

This block is in charge of comparing the output command driven by the output_routing block with the filtered feedback, generating an error_feed signal when a mismatch is detected.

It is necessary to note that this comparison is disabled for disable_window (refer to the LSx Output Registers and the HSx Output Registers sections) number of ck clock cycles after a transition has occurred on the related output signal. This needs to be done to disable the comparison when the feedback has not yet reported the effect of the output transition, due to analog delays in its generation and due to the delays introduced by the digital filter of the input feedback.

This block is in charge of disabling the comparison in given time windows and then performing this comparison generating an error_feed signal according to the error tables disable_switch. These tables have been defined in the diagnosis registers (refer to the LSx Output Registers and the HSx Output Registers sections) to provide full flexibility in the generation of the error signals, allowing to set an error when the command and the feedback are equal or opposite, according to the way the application is designed.

6.15.7.10.10 Error_handler Block

This block is in charge of generating the real four output error signals which is actually treated as an interrupt request by each microcore, setting its program counter to the automatic diagnosis interrupt routine (refer to the Diag_routine_addr register (0x10C and 0x12C) section) at the address where the interrupt handling routine starts. The error_feed conditions can be masked by the related en_halt_x signal controlled by a specific instruction.

This block is also in charge of storing the values of all the diagnosis relevant signals in the Err_ucXchY registers. This has to be done in order to allow the identification of the fault on the actuation stage.

Each of these registers can be reset either by the microcode of the relative microcore (always possible) or by reading them through the SPI (configurable). In the same way, the interrupt request can be cancelled either by the microcode (always possible) or by reading the err_uc register through the SPI (configurable).

These two events (the Err_ucXchY registers reset and the possibility to generate interrupt requests) are not fully linked. It is possible to reset the Err_ucXchY registers (and this automatically re-enables also the possibility to generate interrupt requests), or it is possible to re-enable the possibility to generate interrupt requests without resetting the Err_ucXchY registers.

These different behaviors are controlled with the *rstreg* instruction.

6.15.7.10.11 Err_ucXchY Registers

This is the status register controlled by automatic diagnosis: one for each microcore. This register stores all meaningful information whenever an error condition is detected on any of the pairs (output/feedback) by which the microcore is enabled.

The information stored in the register in regard to the output commands and the related voltage (V_{DS} and V_{SOURCE}) feedbacks.

A cksys_missing (PLL output clock not valid) condition does not trigger the Err_ucXchY to be latched, but if the register is latched, the cksys_missing bit shows the cksys status at the same moment when the automatic diagnosis error occurred. The cksys_missing bit is set when the PLL output clock was not valid at the time the automatic diagnosis error occurred.

Table 207. Err_ucXchY_1 Registers (0x162, 0x164, 0x166, 0x168)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	vds_ ls1	cmd_ hs5	vsrc_ hs5	vds_ hs5	cmd_ hs4	vsrc_ hs4	vds_ hs4	cmd_ hs3	vsrc_ hs3	vds_ hs3	cmd_ hs2	vsrc_ hs2	vds_ hs2	cmd_ hs1	vsrc_ hs1	vds_ hs1
R/W	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Lock	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Reset on read	conf.	conf.	conf.	conf.	conf.	conf.	conf.	conf.	conf.	conf.	conf.	conf.	conf.	conf.	conf.	conf.
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 208. Err_ucXchY_2 Registers (0x163, 0x165, 0x167, 0x169)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved			cksys_ missing	cmd_ ls7	cmd_ ls6	vds_ ls6	cmd_ ls5	vds_ ls5	cmd_ ls4	vds_ ls4	cmd_ ls3	vds_ ls3	cmd_ ls2	vds_ ls2	cmd_ ls1
R/W	-			r	r	r	r	r	r	r	r	r	r	r	r	r
Lock	-			-	-	-	-	-	-	-	-	-	-	-	-	-
Reset on read	-			conf.	conf.	conf.	conf.	conf.	conf.	conf.	conf.	conf.	conf.	conf.	conf.	conf.
Reset	000			0	0	0	0	0	0	0	0	0	0	0	0	0

6.15.7.10.12 Fbk_sens_ucX_chY

This register (one for each microcore) select the feedbacks by which each microcore is enabled (e.g. configures if uc0 ch1 is sensitive to V_{DS} errors on HS1).

Table 209. Fbk_sens_uc0ch1 Register (0x180)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ls6_vds_sens	ls5_vds_sens	ls4_vds_sens	ls3_vds_sens	ls2_vds_sens	ls1_vds_sens	hs5_vsrc_sens	hs5_vds_sens	hs4_vsrc_sens	hs4_vds_sens	hs3_vsrc_sens	hs3_vds_sens	hs2_vsrc_sens	hs2_vds_sens	hs1_vsrc_sens	hs1_vds_sens
R/W	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Reset	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1

Table 210. Fbk_sens_uc1ch1 Register (0x181)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ls6_vds_sens	ls5_vds_sens	ls4_vds_sens	ls3_vds_sens	ls2_vds_sens	ls1_vds_sens	hs5_vsrc_sens	hs5_vds_sens	hs4_vsrc_sens	hs4_vds_sens	hs3_vsrc_sens	hs3_vds_sens	hs2_vsrc_sens	hs2_vds_sens	hs1_vsrc_sens	hs1_vds_sens
R/W	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Reset	1	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0

Table 211. Fbk_sens_uc0ch2 Register (0x182)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ls6_vds_sens	ls5_vds_sens	ls4_vds_sens	ls3_vds_sens	ls2_vds_sens	ls1_vds_sens	hs5_vsrc_sens	hs5_vds_sens	hs4_vsrc_sens	hs4_vds_sens	hs3_vsrc_sens	hs3_vds_sens	hs2_vsrc_sens	hs2_vds_sens	hs1_vsrc_sens	hs1_vds_sens
R/W	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Reset	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0

Table 212. Fbk_sens_uc1ch2 Register (0x183)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ls6_vds_sens	ls5_vds_sens	ls4_vds_sens	ls3_vds_sens	ls2_vds_sens	ls1_vds_sens	hs5_vsrc_sens	hs5_vds_sens	hs4_vsrc_sens	hs4_vds_sens	hs3_vsrc_sens	hs3_vds_sens	hs2_vsrc_sens	hs2_vds_sens	hs1_vsrc_sens	hs1_vds_sens
R/W	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
Lock	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
Reset	0	0	1	0	0	0	1	1	1	1	0	0	0	0	0	0

6.15.7.10.13 Output_routing Block

The output routing block provides the final output_command signals. Each of the output_command signals can be:

- the combination of the microcores requests (refer to the Output_switch_box section)
- one device input flag (refer to the LSx Output Registers and HSx Output Registers sections)
- a free-wheeling command referred to one of the high-side outputs. This option can be selected, only on five outputs, by the control signal fw_auto. Refer to Table 213 for the possible combinations

Table 213. Automatic Freewheeling Pre-driver Association

Freewheeling Pre-driver Output	Related Pre-driver High-side
LS5	HS1
LS6	HS2
LS7	HS3
HS5	HS4
LS4	HS5

6.15.7.10.14 Freewheeling Drive

This block controls the free wheeling output according to the high-side output command, and the fw_auto signals.

- Automatic mode: (enabled when fw_auto is set to 1) the freewheeling output is always the opposite of the high-side output. The drive request by the microcore to the output used for the freewheeling function are neglected (refer to the above table). In this case, the freewheeling command goes high after a programmable time (programmed in HSx Output Registers) with respect to the HS command falling edge. It is also important to note that in this mode the HS command rising edge is always delayed of the same programmable time with respect to the rising edge requested by the microcore. This is done in order to always assure that high-side and freewheeling is never active at the same time.
- Manual mode: (enabled when fw_auto is set to 0) the output that can be used as a freewheeling is not driven by this block. The output is instead driven by the combined microcores requests.Fw_external_request

It is possible to activate automatic freewheeling even when the microcode is not running, by writing the corresponding bit of this register.

Table 214. Fw_ext_req Register (0x16A)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved											ls4_ fw_en	hs5_ fw_en	ls7_ fw_en	ls6_ fw_en	ls5_ fw_en
R/W	-											r/w	r/w	r/w	r/w	r/w
Lock	-											yes	yes	yes	yes	yes
Reset	00000000000											0	0	0	0	0

- ls5_fw_en: if set, the low-side pre-driver 5 is driven as a freewheeling relative to high-side pre-diver 1, otherwise the status is defined by the microcore request (*stfw* instruction)
- ls6_fw_en: if set, the low-side pre-driver 6 is driven as a freewheeling relative to high-side pre-diver 2, otherwise the status is defined by the microcore request (*stfw* instruction)
- ls7_fw_en: if set, the low-side pre-driver 7 is driven as a freewheeling relative to high-side pre-diver 3, otherwise the status is defined by the microcore request (*stfw* instruction)
- ls5_fw_en: if set, the high-side pre-diver 5 is driven as a freewheeling relative to high-side pre-diver 4, otherwise the status is defined by the microcore request (*stfw* instruction)
- ls4_fw_en: if set, the low-side pre-driver 4 is driven as a freewheeling relative to high-side pre-diver 5, otherwise the status is defined by the microcore request (*stfw* instruction)

6.15.7.10.15 Diagnosis_option

It is possible to select the command used for the automatic diagnosis (refer to the Channel_check Block section). The command used to enable the automatic diagnosis can be selected (refer to the Channel_check Block section). If this option bit is set to ‘0’, the command used is the combination of all the sequencer’s requests and their output accesses (refer to the Out_acc_ucX_chY section). It means that the output command is forced by the microcore. If the option is set to ‘1’, the automatic diagnosis is subsequently disabled for all those drivers whose output has been disabled, due to conditions such as DRVEN pin low, undervoltage detected on the VCCP pin, etc (refer to the Out_acc_ucX_chY section).

Table 215. Diagnosis_option Register (0x16B)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															Diag option
R/W	-															r/w
Lock	-															yes
Reset	000000000000000															1

7 CPU Features and Operation

7.1 Introduction

This section describes the features and operation of the microcores (central processing unit, or CPU, and development support functions) used in the 33816 device.

7.2 Features

The 33816 provides a set of two logic channels. A total of four similar microcores are implemented in the two logic channels of the 33816.

Each logic channel consists of:

- Two 16-bit processing units (microcores) that have a specific programming model
- One Code RAM - 1023 x 16-bit. This memory dedicated to microcode storage is shared between the two microcore of logic channel
- One Data RAM - 64 x 16-bit. This memory dedicated to variable storage is shared between the two microcore of a Logic Channel

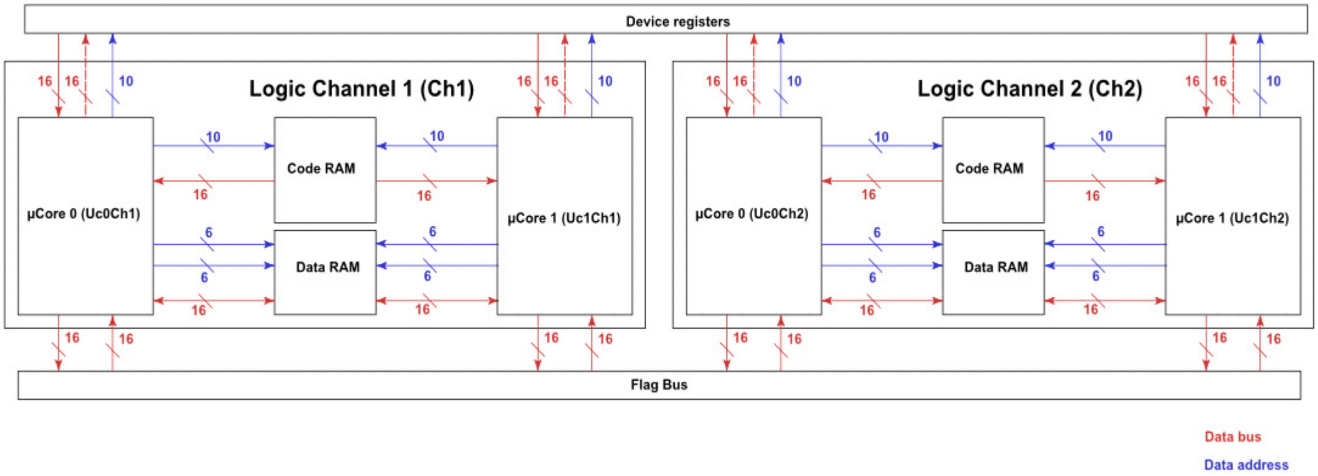


Figure 41. Logic Channels Simplified Block Diagram

Each microcore consists of:

- An instruction decode (Instruction_decoder) that manages all the instructions set
  - The instruction decode includes an internal register multiplexer (Internal_reg_mux) that manages interactions with the memories and the peripheral functions (ALU, counters…)
- .A program counter (Uprogram_counter) that manages the code line to be executed by the instruction decoder. This program counter includes:
  - A program counter register (uPC)
  - An auxiliary register to store the program counter value when handling subroutine
- An Interrupt return register to store the program counter value when handling interrupt.
- One 16-bit ALU
- Four counters

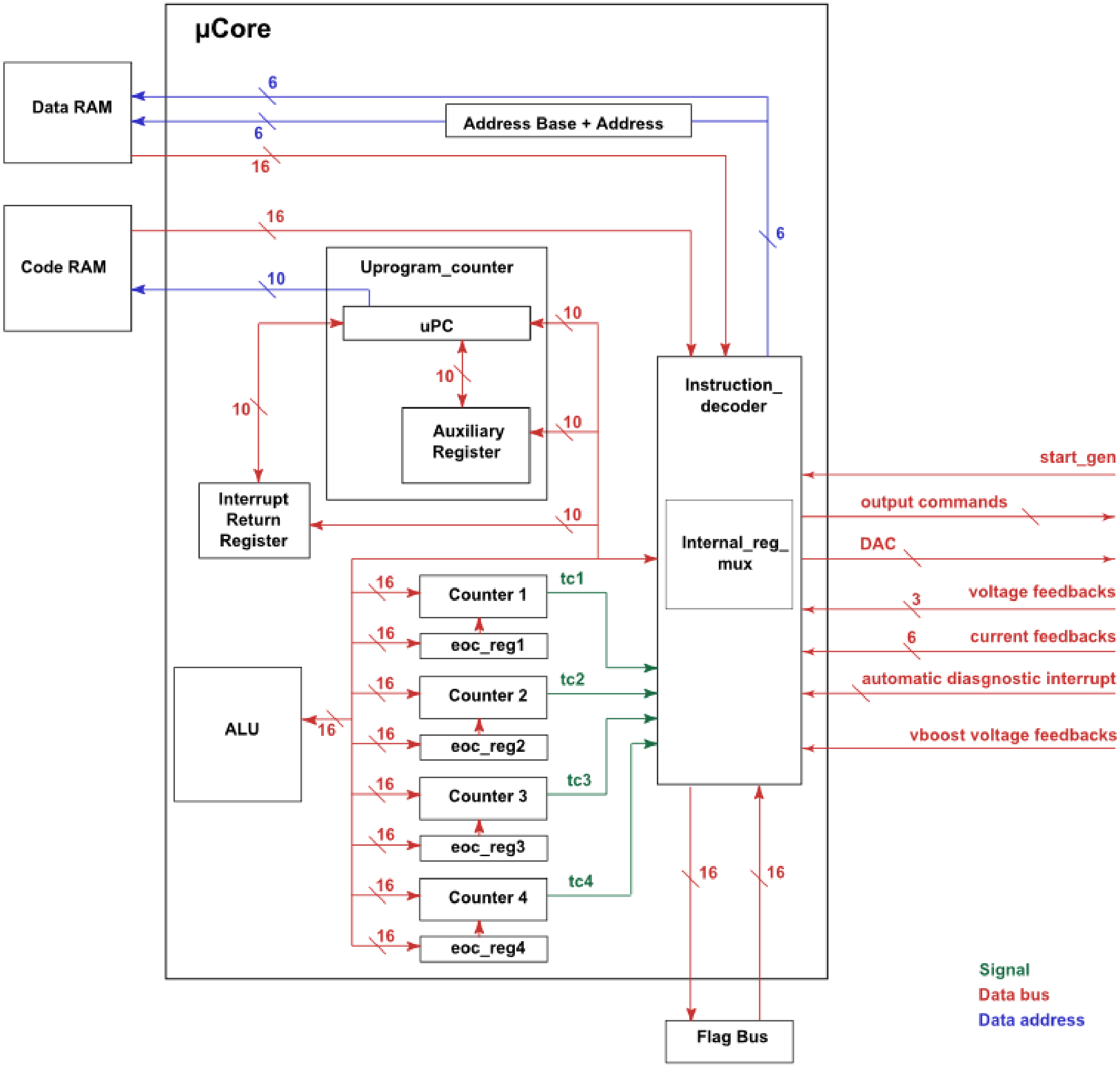


Figure 42. Microcore Block Diagram

7.3 Symbols and Notation

The symbols and notation shown here are used throughout the manual.

7.3.1 Abbreviations for System Resources

iret – Software Interrupt Return Register
aux – Subroutine Auxiliary Register
jr1 – Jump Register 1
jr2 – Jump Register 2
count1 – Counter Register 1
count2 – Counter Register 2
count3 – Counter Register 3

count4 – Counter Register 4
eoc1 – End Of Count Register 1
eoc2 – End Of Count Register 2
eoc3 – End Of Count Register 3
eoc4 – End Of Count Register 4
flag – Flag Register
ctrl_reg – microcore Control Register
status_bits – microcore Status Register
spi_data – SPI Backdoor Register
dac_sssc – DAC Register Same microcore Same Channel
dac_ossc – DAC Register Other microcore Same Channel
dac_ssoc – DAC Register Same microcore Other Channel
dac_osoc – DAC Register Other microcore Other Channel
dac_4h4n – DAC Register 4h and 4n
spi_add – SPI Backdoor Address
irq_status – Interrupt Status Bits Register
ch_rxtx – Other Channel Communication Register
uPC – Program Counter Register
r0 – ALU General Purpose Register 0
r1 – ALU General Purpose Register 1
r2 – ALU General Purpose Register 2
r4 – ALU General Purpose Register 4
ir – ALU Immediate Register
mh – ALU MSB Multiplication Result Register
ml – ALU LSB Multiplication Result Register
arith_reg – ALU Condition Register

7.3.2 Operators

+ - Addition
– - Subtraction
? - Logical AND
+ - Logical XOR (inclusive)
(+) - Logical OR (exclusive)
× - Multiplication
÷ - Division
\ - Negation, logical NOT. One’s complement (invert each bit of a byte or a 16-bit word)
=> - Transfer
<=> - Exchange
>> n - Right shift of n bit(s)
<< n - Left shift of n bit(s)

7.3.3 Definitions

Logic level 1 is the voltage that corresponds to the true (1) state.
Logic level 0 is the voltage that corresponds to the false (0) state.
Set refers specifically to establishing logic level 1 on a bit or bits.
Cleared refers specifically to establishing logic level 0 on a bit or bits.

Asserted means that a signal is in active logic state. An active low signal changes from logic level 1 to logic level 0 when asserted, and an active high signal changes from logic level 0 to logic level 1.

Negated means that an asserted signal changes logic state. An active low signal changes from logic level 0 to logic level 1 when negated, and an active high signal changes from logic level 1 to logic level 0.

RAM is the acronym of Random Access Memory

Code RAM is a 1023 x 16-bit RAM area dedicated to the storage of the microcode. Two microcores share one 64 x 16-bit Data RAM area**.**

Data RAM is a 64 x 16-bit RAM area dedicated to the storage of the variables. Two microcores share one 64 x 16-bit Data RAM area.

LSB means Least Significant Bit or Bits.

MSB means Most Significant Bit or Bits.

A range of bit locations is referred to by mnemonic and the numbers that define the range. For example, ctrl_reg[15:8] is the high byte of the microcore control register.

Microcores are the CPUs integrated in the MC33816.

ALU is the acronym of Arithmetic Logic Unit. The ALU is a part of the microcore is charge of executing the mathematic and logic instructions.

SPI is the acronym of Serial Peripheral Interface. The SPI is the primary communication interface with the application MCU.

MCU is the acronym of Microcontroller Unit. The main MCU is the main digital device of the electronic module.

8 CPU Features and Operation Overview

8.1 Introduction

This section describes the 33816 microcore programming model, register set, the data types used, and basic memory organization. This section is split in three main sub-sections:

- Memory and signals management.
- The ALU
- Data RAM addressing modes

8.2 Memory and Signals Management Programming Model

Table 216. Microcore Programming Model

9	iret	0	INTERRUPT RETURN REGISTER
9	aux	0	AUXILIARY REGISTER
9	jr1	0	JUMP REGISTER 1
9	jr2	0	JUMP REGISTER 2
15	cnt1	0	CONTER REGISTER 1
15	cnt2	0	CONTER REGISTER 2
15	cnt3	0	CONTER REGISTER 3
15	cnt4	0	CONTER REGISTER 4
15	eoc1	0	END OF COUNT REGISTER 1
15	eoc2	0	END OF COUNT REGISTER 2
15	eoc3	0	END OF COUNT REGISTER 3
15	eoc4	0	END OF COUNT REGISTER 4
15	flag	0	FLAG REGISTER
15	ctrl_reg	0	CONTROL REGISTER
15	status_bits	0	STATUS REGISTER
15	spi_data	0	SPI BACKDOOR DATA REGISTER
13	dac_sssc	0	DAC REGISTER SAME UCORE SAME CHANNEL
13	dac_ossc	0	DAC REGISTER OTHER UCORE SAME CHANNEL
13	dac_ssoc	0	DAC REGISTER SAME UCORE OTHER CHANNEL
13	dac_osoc	0	DAC REGISTER OTHER UCORE OTHER CHANNEL
11	dac_4h4n	0	DAC REGISTER 4H AND 4NEG
7	spi_add	0	SPI BACKDOOR ADDRESS REGISTER
15	irq_status	0	INTERRUPT STATUS REGISTER
15	ch_rxtx	0	OTHER CHANNEL COMMINUCATION REGISTER
15	uPC	0	PROGRAM COUNTER
5	add_base	0	ADDRESS BASE REGISTER

8.2.1 Interrupt Return Register (iret)

The interrupt return register (iret) is a 10-bit register that stores the address of the next instruction to be executed when an interrupt return is issued. The current uPC register value is automatically stored in the iret register each time a software interrupt request is called, by means of the *reqi* instruction or when an automatic interrupt is issued.

The uPC register is loaded with the iret register contents when the *iret* instruction is called or an automatic software interrupt return is executed. However, according to the returned behavior set with the *iconf* instruction, the return from the instruction address loaded back into the uPC register can be the microcore Code RAM entry point.

This register cannot be directly accessed by the instruction decoder.

8.2.2 Auxiliary Register (aux)

The auxiliary register is a 10-bit register holding the address of the next instruction to be executed when a return from subroutine is issued. The return from subroutine address is automatically stored in this aux register each time a jump to a subroutine is called, by means of the *jtsf* and *jtsr* instructions.

The uPC register is loaded back with the aux register contents when the *rfs* instruction is executed.

Loading the auxiliary register value to register or DRAM is possible by means of the *cp* and *store* instructions.

8.2.3 Jump Registers (jr1, jr2)

The jump registers 1 and 2 are two 10-bit registers are loaded with the *instructions ldjr1, ldjr2,* and *load* previously to execute jumps far or ‘wait table’ setting instructions (*cwef, jarf, jcrf, jfbkf, jmpf, jocf, joidf, joslf, jsrf, jstf*).

This register contains the destination address when a jump far instruction (absolute jump) is executed.

8.2.4 Counter Registers (cnt1, cnt2, cnt3, cnt4)

Each microcore has its own set of four independent counters. The four registers are 16-bit registers containing the four counter values.

Each time the maximum counter value (0xFFFF) is encountered the counter value is reset (0x0000).

The counter value is incremented by 1 independently for each counter at each ck cycle for the counter 1 and 2. A pre-scaling can be applied to the counter 3 and 4 according to the Counter_34_prescaler registers (0x111, 0x131).

The counter registers can be loaded with the *cp, ldca, ldcd* and *load* instructions.

Loading a counter value to register or DRAM is possible by means of *cp* and *store* instructions.

8.2.5 End Of Count Registers (eoc1, eoc2, eoc3, eoc4)

Each counter is associated with a 16-bit end of count register. Each time the corresponding counter value reaches the end of count value, the corresponding tcx signal is set high and the counter incrementing is stopped.

This tcx signal can be used as a wait table condition.

The end of count registers can be loaded with the *cp* and *load* instructions.

Loading the end of counter value to register or DRAM is also possible by means of *cp* and *store* instructions.

8.2.6 Flag Register (flag)

The flag register controls the of reading of the 16 flags. Each flag bit issued from each microcore is combined (AND). The dominant level is the logic level 0.

The flag states can be read and handled by each microcore with the *cp* and *store* instructions.

The flag values can be set by using the *stf* instruction.

8.2.7 Control Register (ctrl_reg)

The control register is made up of two 8-bit parts:

- ctrl_reg[7:0], can only be read by the microcore while the MCU can read or write this slice. This slice can be used as an input communication channel from an external device.
- ctrl_reg[15:8] have a configurable behavior by means of the Control_register_split registers (0x112 and 0x132). This slice can be used like either the lower slice (ctrl_reg[7:0]) or as the status byte.

The ctrl_reg[15:8] can be written bit-by-bit using the *stcrl* instructions in status mode

8.2.8 Status Register (status_bits)

The status register is 16-bit register. It can be read and written by the microcore, while the MCU can only read this register.

The register can be used as an output communication channel towards an external device or as temporary register. A combination of the two is possible.

The ctrl_reg[15:8] can be written bit-by-bit using the *stcrb* instructions.

Loading a control register value to register or DRAM is also possible by means of *cp* and *store* instructions.

8.2.9 SPI Backdoor Data Register (spi_data)

The SPI backdoor data register is a 16-bit buffer to store the data for transmission through the SPI backdoor or the buffer for the SPI backdoor data reading.

The SPI backdoor data register can be loaded with the *cp* and *load* instructions.

Loading a backdoor data value to the register or DRAM is also possible by means of *cp* and *store* instructions.

8.2.10 DAC Registers (dac_sssc, dac_ossc, dac_ssoc, dac_osoc, dac_4h4n)

The DAC registers are used to setup the current measurement block DACs. These DACs are affected as shown below.

Table 217. Current Measurement DACs Affectation to Microcores

microcore	sssc	ossc	ssoc	osoc
Uc0Ch1	dac1	dac2	dac3	dac4l
Uc1Ch1	dac2	dac1	dac4l	dac3
Uc0Ch2	dac3	dac4l	dac1	dac2
Uc1Ch2	dac4l	dac3	dac2	dac1

The DAC Register contains the DAC value and the offset compensation value to be set of read:

- dac_XsYc[13:8] contains the offset compensation value of the related current measurement block (dac_offset_x).
- dac_XsYc[7:0] contains the offset compensation value of the related current measurement block (dac_value_x).

The dac_4h4n is unique and can be access by all the microcores. This register is split in two slices:

- dac_4h4n[11:8] contains the DAC4neg register value (dac_value_4neg)
- dac_4h4n[7:0] contains the DAC4h register value (dac_value_4h)

These DACs can be set by using the *stdm* instruction to setup the access mode.

The DAC registers can be loaded with the *cp* and *load* instructions.

Loading DAC registers values to other registers or DRAM is also possible by means of *cp* and *store* instructions.

8.2.11 SPI Backdoor Address Register (spi_add)

This 8-bit register defines the SPI backdoor address for read or write access.

The SPI backdoor address register can be loaded with the *cp* and *load* instructions.

Loading the SPI backdoor address register value to other registers or DRAM is also possible by means of *cp* and *store* instructions.

8.2.12 Interrupt Status Register (irq_status)

The interrupt status register is a 16-bit register that is a living copy of the Uc0_irq_status registers (0x10F and 0x12F) and in the Uc1_irq_status registers (0x110 and 0x130).

Loading the interrupt status register value to other registers or DRAM is possible by means of *cp* and *store* instructions.

8.2.13 Channel Exchange Data Register (ch_rxtx)

This 16-bit register is used to exchange data between all 4 microcores. The communication configuration can be setup using the *stcrt* instruction.

The channel exchange data register can be loaded with the *cp* and *load* instructions.

Loading the channel exchange data register value to other registers or DRAM is also possible by means of *cp* and *store* instructions.

8.2.14 Program Counter Register (uPC)

The program counter (uPC) is a 16-bit register that holds the address of the next instruction to be executed. It automatically increments each time an instruction is fetched.

8.2.15 Data RAM Address Base Register (add_base)

This 6-bit register defines the offset to be applied when executing a read or write instruction in the Data RAM. This offset enables using the *Ofs* operand in the instructions *ldcd*, *load* and *store*.

The address base is set by means of the *stab* instruction.

42.3 ALU Programming Model

Table 218. ALU Programming Model

15	r0	0	ALU GENRAL PURPOSE REGISTERS
15	r1	0	
15	r2	0	
15	r3	0	
15	r4	0	
15	ir (r5)	0	IMMEDIATE REGISTER
15	mh (r6)	0	MULTIPLICATION RESULT REGISTER HIGH
15	ml (r7)	0	MULTIPLICATION RESULT REGISTER LOW
15	arith_reg	0	CONDITION REGISTER

8.2.16 ALU general Purpose Registers (r0, r1, r2, r3, r4)

The 16-bit ALU general purpose registers are used in most of the logic and arithmetic operations as sources and/or destination registers.

8.2.17 ALU Immediate Register (ir)

The immediate register is used as the source register for logic mask operations (AND, NOT, OR, XOR) and can be used as an ALU general purpose register.

8.2.18 ALU Multiplication Result Registers (mh, ml – reg32)

The ALU multiplication result registers mh and ml are mainly used as destination registers for multiplication operations or for shift register operations.

In these two cases, the 32-bit ALU multiplication result register reg32 is used. This register is formed by the concatenation of the two multiplication results registers mh and ml.

Table 219. Reg32 Register

<table>
<tr><td colspan="2">31 Reg32 0</td><td rowspan="2">MULTIPLICATION RESULT REGISTER</td></tr>
<tr><td>15 mh 0</td><td>15 ml 0</td></tr>
</table>

The 16-bit ALU general purpose registers mh and ml can be used as ALU general purpose registers.

8.2.19 Condition Register (arith_reg)

The 16-bit ALU condition register contains the ALU values condition and the flag issued from the ALU operations.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SB	CS	C	A1	A0	MN	MM	MO	ML	RZ	RS	UU	UO	SU	SO	OD

SB - Shift out bit: the last bit shifted out (either left or right) from a shift operation.

CS - Last conversion sign: the product of all signs removed by the toint instruction. This bit can be reset by set the Rst operand of the toint instruction.

C – Carry over bit: the carry produced by the last addition or subtraction operation.

A1 A0 – Arithmetic logic: the parameter used for addition and subtraction operations. It has four possible values:

- ‘00’ or ‘10’: no limitation is imposed on addition or subtraction. In case of an overflow, the result should be represented on 17 bits, but only the 16 LSBs of this result are available in the target register. In case of an underflow, the result stored in the target register is ‘65536 - the correct result’, which can always be represented on 16 bits.
- ’01’: the result of addition or subtraction are saturated between the maximum possible value (if overflow) or the minimum possible value (if underflow). The numbers are considered to be twos-complement format, so saturated between 0x8000 (-32768) and 0x7FFF (+32767).
- ’11’: the result of addition or subtraction are saturated between the maximum possible value (if overflow) or the minimum possible value (if underflow). The numbers are considered to be unsigned so saturated between 0xFFFF (65535) and 0x0000 (0).

MN - Mask result is 0x0000: set if the result of the last mask operation is 0x0000.

MM - Mask result is 0xFFFF: set if the result of the last mask operation is 0xFFFF.

MO - Multiplication or shift overflow: cleared if the 16 MSBs of the last multiplication or 32-bit shift result are all ‘0’, otherwise set to ‘1’.

ML - Multiplication or shift precision loss: cleared if the 16 LSBs of the last multiplication or 32-bit shift result are all ‘0’, otherwise set to ‘1’.

RZ - Addition or subtraction result is zero: the result of the last addition or subtraction is zero.

RS - Addition or subtraction result is negative: the result of the last addition or subtraction is negative.

UU - Unsigned underflow: set if the last addition or subtraction produced underflow, considering the operands as unsigned numbers.

UO - Unsigned overflow: set if the last addition or subtraction produced overflow, considering the operands as unsigned numbers.

SU - Signed underflow: is set if the last addition or subtraction produced underflow, considering the operands as two’s complement numbers.

SO - Signed overflow: set if the last addition or subtraction produced overflow, considering the operands as two's complement numbers.

OD - Operation complete: set by the ALU. The Instruction_decoder can only read it. This bit is set to '0' when a multi-cycle operation is in progress, otherwise is set to '1'. If an ALU operation is issued when another operation is in progress (in that case the OD bit is set to '0'), the request is neglected.

8.3 Addressing Modes

8.3.1 Immediate Addressing Mode (IM)

The immediate addressing mode is used to immediately access the Data RAM areas by the instruction. In this addressing mode, there is no need to pre-load an address in any register as the address is an operand of the instruction.

Example:

load *0 r0 _ofs*;

In this example, the data contained into the address *0* of the data RAM is loaded into the ALU register *r0*.

8.3.2 Direct Addressing Mode (DM)

The direct addressing mode is used to directly access the microcore registers by the instruction. In this addressing mode, there is no need to pre-load addresses in any register as the address is a predefined operand the instruction.

The pre-defined labels that are use in this mode are listed in the section **Operand Subsets**.

Example:

cp *r0 r1*;

In this example, the data contained in the ALU register *r0* is copied in the ALU register *r1.*

The assembler replaces the operand label by the corresponding binary value.

8.3.3 Extended Addressing Mode (EM)

The extended addressing mode requires pre-loading the address in a dedicated register. This addressing mode is mainly used to access Code RAM lines.

Example:

ldjr1 *20;*

jmpf *jr1*;

In this example, the jump register *jr1* is first loaded with the Code RAM address where the uPC counter is expected to go.

The *jmpf* instruction is executed and the uPC counter is handled such as to jump to the expected Code RAM address.

8.3.4 Indexed Addressing Modes (XM)

The indexed addressing mode is used to immediately access the Data RAM areas by the instruction using an offset. In this addressing mode, loading the offset value is required prior to use the indexed addressing mode.

Example:

stab *7;*

load *1 r1 ofs*;

In this example, an offset value of 7 is first loaded into the address base register add_base. The data contained in the Data RAM address 1 (+7) is loaded into the ALU register *r1*. The Ofs operand is set to *ofs* to enable the indexed Data RAM addressing.

8.3.5 Relative Addressing Modes (RM)

The relative addressing mode is used to access the Code RAM line in the range of -16 to +15 lines referenced to the instruction line executed.

Example:

jmpr *6;*

In this example, next instruction executed is located 6 lines below the Code RAM line currently executed. The operant used for relative addressing is a two's complement format number

Table 220. Complement Format Description

two's complement relative value (binary)					Relative Address Value (decimal)	Relative Destination Position (decimal)
MSB (sign bit)				LSB		
0	1	1	1	1	0x0F	+15
0	0	0	0	1	0x01	+1
0	0	0	0	0	0x00	0
1	1	1	1	1	0x1F	-1
1	0	0	0	0	0x10	-16

The relative address can be replaced by an univocal label. The compiler replaces the label by the suitable number. The targeted line must in range of 16 to +15 lines referenced to the instruction line executed.

9 Instruction Set Overview

9.1 Introduction

This section contains general information about the central processor unit instruction set. Its description is organized by function group.

9.2 Instruction Set Description

Table 221. 33816 Instruction Set Overview

Instruction Name	Instruction Description
add	Two ALU registers addition to ALU register
addi	ALU register addition with immediate value to ALU register
and	AND-mask on ALU register with the immediate register to ALU register
bias	Enable high-side and low-side bias
chth	Change V_{DS} and V_{SRC} threshold
cp	Copy source register data in destination register
cwef	Create wait table entry far
cwer	Create wait table entry relative
dfcsct	Define current shortcut
dfsct	Define pre-driver output shortcuts
endiag	Enable automatic diagnosis
endiaga	Enable all automatic diagnosis
endiags	Enable automatic diagnosis shortcuts
iconf	Interrupt configuration
iret	Return from interrupt
jarf	Jump far on arithmetic condition
jarr	Jump relative on arithmetic condition
jcrf	Jump far on control register condition
jcrr	Jump relative on control register condition
jfbkf	Jump far on feedback condition
jfbkr	Jump relative on feedback condition
jmpf	Unconditional jump far
jmpr	Unconditional jump relative
jocf	Jump far on condition
jocr	Jump relative on condition
joidf	Jump far on microcore condition
joidr	Jump relative on microcore condition
joslf	Jump far on start condition
joslr	Jump relative on start condition

Table 221. 33816 Instruction Set Overview (continued)

Instruction Name	Instruction Description
jsrf	Jump far on status register bit condition
jsrr	Jump relative on status register bit condition
jtsf	Jump far to subroutine
jtsr	Jump relative to subroutine
ldca	Load counter from ALU register and set outputs
ldcd	Load counter from Data RAM and set outputs
ldirh	Load 8-MSB ir register
ldirl	Load 8-LSB ir register
ldjr1	Load jump register 1
ldjr2	Load jump register 2
load	Copy data from Data RAM to register
mul	Two ALU registers multiplication to multiplication result register
muli	ALU register multiplication with Immediate value to multiplication result register
not	Invert ALU register bits
or	OR-mask on ALU register with immediate register to ALU register
rdspi	SPI read request
reqi	Software interrupt request
rfs	Return from subroutine
rstreg	Registers reset
rstsl	Start-latch registers reset
sh32l	Shift left multiplication result register
sh32li	Shift left multiplication result register of immediate value
sh32r	Shift right multiplication result register
sh32ri	Shift right multiplication result register of immediate value
shl	Shift left ALU register
shl8	Shift left ALU register of 8 bits
shli	Shift left the ALU register of immediate value
shls	Shift left signed ALU register
shlsi	Shift left signed ALU register of immediate value
shr	Shift right ALU register
shr8	Shift right ALU register of 8 bits
shri	Shift right the ALU register of immediate value
shrs	Shift right signed ALU register
shrsi	Shift right signed ALU register of immediate value
slab	Select Data RAM address base
slfbk	Select HS2/4 feedback reference
slsa	Select SPI address
stab	Set Data RAM address base

Table 221. 33816 Instruction Set Overview (continued)

Instruction Name	Instruction Description
stadc	Set ADC mode
stal	Set arithmetic logic mode
stcrb	Set control register bit
stcrt	Set channel communication register
stdcctl	Set DC-DC control mode
stdm	Set DAC registers mode access
stdrm	Set Data RAM read mode
steoa	Set end of actuation mode
stf	Set flag
stfw	Set freewheeling mode
stgn	Set current measure operational amplifier gain
stirq	Set IRQB pin
sto	Set single pre-driver output
stoc	Set offset compensation
store	Store register data in Data RAM
stos	Set pre-driver output shortcuts
stslew	Set pre-driver output slew-rate mode
stsrb	Set status register bit
sub	Two ALU registers subtraction to ALU register
subi	ALU register subtraction with immediate value to ALU register
swap	Swap bytes inside ALU register
toc2	Integer to two’s complement conversion in ALU register
toint	two’s complement to integer conversion in ALU register
wait	Wait until condition satisfied
wrspi	SPI write request
xor	XOR-mask on ALU register with the immediate register to ALU register

9.3 Arithmetic Logic Unit (ALU) Instructions

9.3.1 Addition and Subtraction Instructions

The addition (*add, addi*) and subtraction (*sub, subi*) operations allow handling of 16-bit numbers.

One of the ALU register can be added or subtracted to another ALU register or to a 4-bit immediate value according to the instruction used. The result is always stored in an ALU register.

The ALU registers called by the instruction operands must be previously loaded using the *load* and *cp* instructions.

Carry, result is zero, result is negative, overflow and underflow are reported in the ALU condition register arith_reg:

- C – Carry over bit
- RZ - Addition or subtraction result is zero
- RS - Addition or subtraction result is negative
- UU - Unsigned underflow
- UO - Unsigned overflow

- SU - Signed underflow
- SO - Signed overflow.

The addition or subtraction result can be unsigned or signed according to the A1 and A0 bit of the ALU configuration register. arith_reg.

The addition and subtraction instructions require one ck clock cycle to be executed.

9.3.2 Multiplication Instructions

The multiplication instructions (*mul, muli*) operation allows handling of 16-bit numbers.

The ALU registers called by the instruction operands must be previously loaded using the *load* and *cp* instructions.

According to the instruction used one of the ALU register can be multiplied by another ALU register or by a 4-bit immediate value. The result is always stored in the ALU multiplication result register reg32.

Overflow, loss of precision, operation complete are reported in the ALU Condition register arith_reg:

- MO - Multiplication shift overflow
- ML - Multiplication shift precision loss
- OD – Operation complete

The multiplication instructions require 17 ck clock cycles to be executed.

The OD bit of the ALU condition register is low and the result is unavailable until the multiplication is completed.

The ALU is not available and the ALU instructions are ignored until the shift operation is completed except for the *stal* instruction.

9.3.3 Mask Instructions

The 33816 ALU offers the possibility to apply logic mask on the data stored into the ALU registers.

The ALU register must be previously loaded with the 16–bit source value using the *load* and *cp* instructions.

According to the instruction, the source value can be ANDed, NOTed (invert), ORed or XORed with the immediate register ir 16-bit value.

The result is always available in the 16–bit source register.

The mask instructions (*and, not, or, xor*) require one ck clock cycle to be executed.

9.3.4 Shift Instructions

The shift instructions allow single or multiple left and right shifts. This instruction subset allows the following ALU registers shift:

- Left shift of the multiplication result register reg32 with a 16-bit ALU register (sh32l)
- Right shift of the multiplication result register reg32 with a 16-bit ALU register (sh32r)
- Left shift of the multiplication result register reg32 with a 4-bit immediate value (sh32li)
- Right shift of the multiplication result register reg32 with a 4-bit immediate value (sh32ri)
- Left shift of a 16-bit ALU register with another a 16-bit ALU register (shl)
- Right shift of a 16-bit ALU register with another a 16-bit ALU register (shr)
- Left shift of a 16-bit ALU register of 8 positions (shl8)
- Right shift of a 16-bit ALU register of 8 positions (shr8)
- Left shift of a 16-bit ALU register with a 4-bit immediate value (shli)
- Right shift of a 16-bit ALU register with a 4-bit immediate value (shri)
- Left shift of a 16-bit ALU register containing a signed value with another a 16-bit ALU register (shls)
- Right shift of a 16-bit ALU register containing a signed value with another a 16-bit ALU register (shrs)
- Left shift of a 16-bit ALU register containing a signed value with a 4-bit immediate value (shlsi)
- Right shift of a 16-bit ALU register
- containing a signed value with a 4-bit immediate value (shrsi)

The shift operation result is always the source register.

Overflow, loss of precision, operation complete are reported in the ALU Condition register arith_reg:

- SO – Shift out bit
- MO - Multiplication shift overflow

- ML - Multiplication shift precision loss
- OD – Operation complete

The number of ck clock cycle required to complete the operation is equal to the shift value.

Example:

Sh32i *4;*

In this example 4 ck clock cycles are required to complete the operation.

The OC bit of the ALU condition register is high and the result is unavailable until the shift operation is completed.

The ALU is not available and the ALU instructions are ignored until the shift operation is completed except for the *stal* instruction.

Table 222. Complement Format Description

Instruction Name	Source Register	Shift value Register/ Value	Result Register	Number of ck Clock Cycles to Complete Operation	Number of Preliminary Register Loading Instructions.
sh32 l/ sh32r	ALU multiplication result register reg32	16-bit ALU register	ALU multiplication result register reg32	[16-bit ALU register] value	3
sh32li/sh32ri	ALU multiplication result register reg32	4-bit immediate value	ALU multiplication result register reg32	4-bit immediate value	2
shl/shr	16-bit ALU register	16-bit ALU register	16-bit ALU register	[16-bit ALU register] value	2
shl8/shr8	16-bit ALU register	‘8’	16-bit ALU register	1	1
shli/shri	16-bit ALU register	4-bit immediate value	16-bit ALU register	4-bit immediate value	2
shls/shrs	16-bit ALU register	16-bit ALU register	16-bit ALU register	[16-bit ALU register] value	2
shlsi/shrsi	16-bit ALU register	4-bit immediate value	16-bit ALU register	4-bit immediate value	1

9.3.5 Swap Instruction

The *swap* instruction offers the possibility to swap the high-byte and the low-byte of the data contained into an ALU register. The result is stored in the source register.

The swap operation requires one ck clock cycle to be executed.

9.3.6 Format Conversion

The ALU offers the possibility to convert:

- an unsigned value (integer) to a signed value (two’s complement) using the *toc2* instruction
- a signed value (two’s complement) to an unsigned value
- (integer) using the *toint* instruction

The *Toc2* instruction sets the MSB according to the bit CS - Last conversion sign of the ALU condition register.

The CS - Last conversion of the ALU condition register can be changed using the *toint* instruction. The CS bit is XORed with the existing operand MSB (the operand that contains the value to be handled) or replace by the operand MSB according to the *Rst* operand.

The conversion operations require one ck clock cycle to be executed.

9.4 Configuration Instructions

9.4.1 Pre-drivers

9.4.1.1 Pre-driver Output Shortcuts

The 33816 provides an enhanced way to manage the output pre-drivers by using shortcuts. These shortcuts are defined by the instruction *dfsct*. This instruction defines a set of 3 shortcuts selected among all the low-side and high-side pre-driver outputs (hsx_command, lsx_command).

The instruction *dfsct* requires one ck clock cycle to be executed.

9.4.1.2 Pre-driver Outputs Actuation

Each of the low-side and high-side pre-drivers output can be directly and individually actuated by means of the *sto* instruction. This function doesn't require a shortcut definition.

A second way to enable the pre-drivers outputs is the *stos* instruction that allow actuating up to three output simultaneously. The instruction required a preliminary definition of the shortcuts by means of the *dfcsct* instruction.

The pre-drivers output actuation is possible only the microcore running the related instructions is allow to control the pre-drivers outputs. These output commands access is granted by the setting the appropriate bits in the Out_acc_ucX_chY (0x184, 0x185, 0x186, 0x187) configuration registers.

Both *sto* and *stos* instruction require one ck clock cycle to be executed.

9.4.1.3 Pre-drivers Output Slew rate

Each of the low-side and high-side pre-drivers output slew rates can be globally forced using the *stslew* instruction:

- Either to their maximum slew rate
- or to their normal slew rates individually defined in the Hs_slewrate (0x18E) and Ls_slewrate (0x18F) Registers.

The *stslew* instruction requires one ck clock cycle to be executed.

9.4.1.4 End of Actuation

The *steoa* instruction allows to enable the End of Actuation mode. This instruction perform a selective disablement of the bootstrap such as the final decay current slew rate is not disturbed by a 'parasitic' bootstrap capacitor loading.

The *steoa* instruction requires one ck clock cycle to be executed.

9.4.1.5 Bias Structures

The 33816 provides biasing structures that are used for automatic diagnosis functions. Each high-side driver has an individual pull-up voltage source SRCpux connected to V_{CC5} voltage. The high-side drivers 2 and 4 have a specific option to increase the voltage source current capability.

Each low-side drive, except the low-side pre-driver seven, has an individual pull-down voltage source SRCpdx connected to the device ground.

The *bias* instruction allows switching on or off all the biasing structures of the low-side and high-side together. Note that some or all the bias structure of the low-side and high-side pre-drivers can be switched together at the same time either on or off. That means that, for example, switch on the bias structure for the low-side 1 and switch off the bias structure for the low-side 2 using a unique *bias* instruction is not possible.

In such a case, the *bias* instruction must be called two times.

The bias instruction has an effect in the load biasing structure only if the microcore is allowed to control the corresponding pre-driver. Pre-driver individual control is granted by the Out_acc_ucX_chY registers (0x184, 0x185, 0x186, 0x187).

9.4.1.6 DC-DC Mode

Whatever the DC-DC converter regulation mode used (‘Variable frequency’ of ‘Fixed frequency’), the low-side pre-driver seven is periodically directly controlled by the device based on the current measured on the current measurement block four. This automatic mode allows maintaining the current going through the sense resistor between two threshold defined by the DAC4h_value register (0x1A2) and DAC4l_value register (0x1A1).

This automatic current regulation mode (asynchronous mode) is activated by means of the *stdcctl* instruction. When this automatic mode is deactivated (synchronous mode), the low-side pre-driver seven is directly controlled by the microcore.

9.4.2 VDS and VSRC Monitoring

9.4.2.1 Comparator Voltage Threshold

The 33816 provides integrated V_{DS} and V_{SRC} comparators used for driver diagnosis in idle mode.

All five high-side drivers integrate individual V_{DS} (differential voltage MOSFET drain to source) and V_{SRC} (differential voltage MOSFET source to ground) comparators to an internal 3-bit DAC value defined in the Vds_threshold_hs register (0x18A) and the Vsrc_threshold_hs register (0x18B).

Six of the seven low-side pre-driver drivers integrate individual V_{DS} (differential voltage MOSFET drain to source) comparators to an internal 3-bit DAC value defined in the Vds_threshold_ls_1 and Vds_threshold_ls_2 registers (0x18C, and 0x18D). The V_{DS} monitoring is not implemented into seventh low-side pre-diver.

The DAC threshold can be individually set by each microcore using the *chth* instruction. This instruction requires one clock cycle to be executed.

9.4.2.2 High-side Drain Reference Selection for VDS Monitoring

The high-side driver drain for the High-side pre-drivers 2 and 4 can be selected among two voltages reference by means of the *slfbk* instruction:

- The VBATT pin
- The VBOOST pin

This instruction requires one clock cycle to be executed.

9.4.3 Freewheeling Mode

The 33816 can manage MOSFET in freewheeling mode. High-side and low-side pre-drivers association is fixed.

Table 223. Automatic Freewheeling Pre-driver Association

Freewheeling Pre-driver Output	Related High-side Pre-driver
LS5	HS1
LS6	HS2
LS7	HS3
HS5	HS4
LS4	HS5

Two modes can be distinguished:

- Manual freewheeling
- Automatic freewheeling

The mode can be selected by means of the *stfw* instruction. This instruction requires one clock cycle to be executed.

9.4.4 Current Measurement Blocks Configuration

9.4.4.1 Current Comparator Output Shortcuts

The 33816 provides an enhanced way to manage the current comparator outputs by using shortcuts.

The instruction *dfcsct* defines a set of 3 shortcuts dedicated to the four current comparator outputs (cur1_fbk, cur2_fbk, cur3_fbk, cur4l_fbk).

Table 224. Correspondence Between Current Comparator Output Signal and DAC Name

Current feedback signal	DAC Name (operand Name)
cur1_fbk	dac1
cur2_fbk	dac2
cur3_fbk	dac3
cur4l_fbk	dac4l

The *dfcsct* instruction requires one clock cycle to be executed.

9.4.4.2 Current Feedback

The 33816 provides the possibility to connect each of the four microcores to the current feedback signals issued from each current measurement blocks.

Microcode enablement by the current feedback signal is granted by the registers Curr_block_access_1 (0x188) and the Curr_block_access_2 (0x189).

Each of the four main current feedback signals (cur1_fbk, cur2_fbk, cur3_fbk and cur4l_fbk) must be accessed by the microcore to allow interrupt trig on the current threshold or any instruction (wait, jump) using the current feedback signals.

9.4.4.3 Operation Amplifier gain Setting

The differential amplifier gain of each current measurement block can be set among four values by means of the *stgn* instruction.

The *stgn* instruction requires one clock cycle to be executed.

9.4.4.4 Current Offset Compensation

The automatic offset compensation of each of the current measurement operation amplified can be enabled by means of the *stoc* instruction by any of the four microcores.

The *stoc* instruction requires one clock cycle to be executed.

9.4.4.5 DAC Current Feedback Shortcut

The 33816 provides the possibility to each of the four microcores to access and modified the DAC value of each of the current measurement block comparator.

Prior to change one of the 4 DAC value (dac1_value, dac2_value, dac3_value and dac4l_value) the DAC current shortcut must be set by the *dfcsct* instruction.

9.4.4.6 DAC Register Access Control

The *stdm* instruction allows defining and limiting the control (read and write) of:

- the DAC value and Offset compensation for each of the four current measurement blocks
- the DAC4h value, DAC4neg value and the boost
- DAC value

The *stdm* instruction requires one clock cycle to be executed.

9.4.4.7 ADC Mode

The current measurement blocks can be optionally configured as 8-bit analog to digital converters by means of the *stadc* instruction. Each microcore can set all four current measurement blocks.

The *stadc* instruction requires one clock cycle to be executed.

9.5 Digital Control

9.5.1 Start-latch Registers reset

The *rstsl* instruction allows to reset the start_latch_ucx register assigned to the microcore executing the instruction.

The *rstsl* instruction requires one clock cycle to be executed.

9.5.2 Data RAM Access Mode

The *stdrm* instruction offers the possibility to mask the read access to the Data RAM and even to swap the high-byte and low-byte at read (does not affect the Data RAM value, only the value in the destination register when reading Data RAM).

The *stdrm* instruction requires one clock cycle to be executed.

9.5.3 Data RAM Address Base

The 33816 offers the possibility to select the address base for the Data RAM read or write access by means of the *slab* instruction. Either the value contained in the add_base register or in the ALU ir register can be used as address base when accessing the Data RAM in XM mode.

The add_base register value can be set using the *stab* instruction. This operand of this instruction Is the value to be loaded in the add_base register so no preliminary register loading is required.

Both *slab* and *stab* instructions require one clock cycle to be executed.

9.5.4 Flags Control

The bit of the 33816 internal flag bus can individual be set high or low using the *stf* instruction.

The *stf* instruction requires one clock cycle to be executed.

9.5.5 Status and Control registers

9.5.5.1 Registers Reset

The control registers (Ctrl_reg_ucX (0x101, 0x102, 0x121, 0x122)), status registers (Status_reg_ucX (0x105, 0x106, 0x125, 0x126)) and automatic diagnosis registers (Err_ucXchY (0x162 to 0x169)) can be reset individually or in group by means of the *rstreg* instruction.

The *rstreg* instruction requires one clock cycle to be executed.

9.5.5.2 Control and Status Registers

The control and status registers bits can be individually set using the *stcrb* and *stsrb* instructions.

Both *stcrb* and *stsrb* instructions require one clock cycle to be executed.

9.5.6 ALU configuration Register

The ALU of each microcore can be set up individually to handle the saturation behavior when the ALU handles a number exceeding the destination register capacity. This configuration is possible using the *stal* instruction.

The *stal* instruction requires one clock cycle to be executed.

9.5.7 Channel Communication

Each microcore has access to a specific register ch_rxtx dedicated to information sharing between each microcore. The *stcrt* instruction set which microcore ch_rxtx register is being accessed.

Example:

The Uc0Ch1 executes the following code

cp *ir rxtx;*

The Uc0Ch1 executes the following code

stcrt *ossc;*

cp *rxtx sr;*

In this example, the value contained in the ir register of the microcore Uc0Ch1 is copied in the ch_rxtx register of the same microcore. Then the microcore Uc1Ch1 accesses the ch_rxtx register of the microcore Uc0Ch1 and copies the value in its own status register.

The *stcrt* instruction requires one clock cycle to be executed.

9.5.8 Interrupt configuration

Before calling, the microcode interruption routine must be configured using the *iconf* instruction. This instruction is required to determine the automatic return from interrupt behavior.:

- The code execution continues where the main routine was stopped when the automatic return from interrupt conditions are satisfied.
- The code restarts from the entry point defined in the Uc0_entry_point(0x10A, 0x12A) and Uc1_entry_point(0x10B, 0x12B) register when the automatic return from interrupt conditions are satisfied.
- The microcore ignores any automatic return from interrupt.

The automatic return from interrupt conditions are determined by the iret_en bit of the Driver_config register (0x1C5):

- If iert_en is set to ‘0’, the automatic return from interrupt is generated when the pre-drivers are enabled after disable conditions.
- If iret_en is set to ‘1’, the automatic return from interrupt is generated when the Driver_status register (0x1D2) is cleared. In this case, the Driver_status register must be reset on read by setting the driver_enable_rb bit in the Reset_behavior register (0x1CE).

The *iconf* requires one clock cycle to be executed.

9.5.9 SPI Handling Instructions

The 33816 provides a way for each microcode to directly access the SPI registers through the SPI backdoor.

Read and write access can be performed, but is limited by the lock function of each SPI register.

Prior any SPI backdoor data transfer, the SPI address must be specified by means of the *slsa* instruction. This instruction sets the address in the spi_add register.

The SPI write data must be loaded into the spi_data register using *load* or *cp* instructions. The SPI write can then be executed using the *wrspi* instruction.

A SPI backdoor read action can be executed by means of the *rdspi* instruction. The result is available in the spi_data register.

The *slsa* requires one clock cycle to be executed while the *rdspi* and *wrspi* instructions require two clock cycles.

9.5.10 External Interrupt Request

The 33816 IRQB pin has the primary function to report a hardware interrupt to the application MCU. The logic level of this pin can be directly managed by microcode using the stirq instruction. This function overwrites the current state of the IRQB pin.

The *stirq* requires one clock cycle to be executed.

9.6 Diagnosis Instructions

The 33816 integrates an automatic diagnosis feature that can trigger a diagnosis interrupt. The interrupt is triggered according to:

- the V_{DS} feedback signal (hsx_vds_fbk) state, the V_{Src} feedback signal (hsx_src_fbk) state and pre-driver command (hsx_command) state for each of the high-side pre-driver.
- the V_{DS} feedback signal (lsx_vds_fbk) state and pre-driver command (hsx_command) state for each of the low-side pre-driver.

The automatic diagnosis triggers an interrupt according to the combination of the above signals. This combination for the high-side pre-drivers can be set through the Hsx_diag_config_2 (0x154, 0x157, 0x115A, 0x15D, 0x160) registers.

The combination for the low-side pre-drivers can be set through Lsx_diag_config2 (0x141, 0x144, 0x147, 0x14A, 0x14D, 0x150) registers.

The automatic diagnosis can be enabled by two methods:

- Either by direct enablement for a single (*endiag* instruction) or all (*endiaga* instruction) the V_{DS} and V_{SRC} monitoring.
- Or by using the pre-driver shortcuts with the *endiags* instruction. In that case, the pre-driver shortcuts must be previously configured.

The start address of the automatic diagnosis interrupt is set in the Diag_routine_address register (0x10C and 0x12C).

The diagnosis enablement is effective for the pre-driver output configured to the driven by the related microcore. Pre-driver individual control is granted by the Out_acc_ucX_chY registers (0x184, 0x185, 0x186, 0x187).

The biasing circuitry must be configured to make the S_HSx and D_LSx pin biased in idle phases.

The diagnosis instructions (*endiag, endiaga, endiags*) require one clock cycle to be executed.

9.7 Flow Control Instructions

9.7.1 Subroutines

The 33816 instruction set offers the possibility to manage execution of subroutines.

The subroutine is called using the *jtsf* and *jtsr* instructions and the current main routine uPC value is automatically stored in the aux register.

Only one level of subroutine is supported inside or outside an interrupt.

9.7.1.1 Return from Subroutine

Return from subroutine is generated by means of the *rfs* instruction. In this case, the address stored into the aux register is transferred back to the uPC register to allow continuing the main routine execution.

9.7.2 Interrupt

The 33816 instruction set offers the possibility to manage interrupts. Three kinds of interrupts are available:

- automatic diagnosis interrupt (higher priority)
- driver disabled interrupt
- software interrupt (lower priority)

The automatic diagnosis interrupt and driver disable interrupts are triggered according to the corresponding configuration registers:

- Err_ucXchY registers (0x162 to 0x169) for the automatic diagnosis interrupt
- Driver_status register (0x1D2) for the disabled drivers interrupt.

The software interrupt is called by means of the *reqi* instruction.

When an interrupt is requested, the uPC register value is automatically stored in the iret register.

Only one level of interrupt is supported. The other interrupts requested during the initial interrupt execution are queued and are executed in series, or the queue can be cleared according to the *Rst* operand of the *iret* instruction.

9.7.2.1 Return from Interrupt

The return from interrupt is atomically executed or requested by means of the *iret* instruction. Two return from interrupt behavior are possible when using *iret* instruction:

- The address stored into the iret register is transferred back to the uPC register to allow continuing the main routine execution.
- The execution restarts from the entry point address.

If a wait or a conditional jump instruction was interrupted, the return address is defined, restoring the status of the feedbacks at the moment the interrupt.

The *iret* instruction requires one clock cycle to be executed.

9.7.3 Jumps

Branch instructions cause execution flow to change when specific pre-conditions exist. The 33816 instruction set includes:

- Conditional jump relative
- Conditional jump far
- Unconditional jump relative
- Unconditional jump far

Conditional jump instructions have two execution cases:

- The jump condition is satisfied, and a change of flow takes place.
- The jump condition is not satisfied, and no change of flow occurs.

9.7.3.1 Conditional Jump Relative

The “not-taken” case for conditional jump relative is simple. Since the instructions consist of a single word containing the 6-bit relative destination address, the execution continues with the next instruction.

The “taken” case for conditional jump relative instructions require that the uPC register be refilled so execution can continue at a new address. First, the effective address of the destination is calculated using the relative offset in the operand, then the address is loaded into the program counter (uPC).

The conditional jump relative instructions (*jarr, jcrr, jfbkr, jocr, joidr, joslr, jsrr, jtsr*) require one ck clock cycle to be executed in both cases.

9.7.3.2 Conditional Jump Far

The execution of the conditional jump far instructions requires two steps.

The first step consists of loading the destination address in the 10-bit jump register jr1 or jr2 with the *load* and *cp* instructions.

The conditional jump far instruction can then be executed. The two instruction step can be consecutive. The jump instruction is not destructive for the jump register. The data that contains the jump registers can be reused by another instruction.

The “not-taken” case for conditional jump relative is simple. Since the destination is preloaded into the jump registers and the jump register selection is an instruction operand, the execution continues with the next instruction.

In the “taken” case, the effective address of jump is calculated using the10-bit absolute address previously loaded in the appropriate jump register. The suitable jump register is selected by an instruction operand among the two jump registers jr1 and jr2. The address is loaded into the program counter (uPC) and the execution can continue at a new address.

The conditional jump far instructions themselves (*jarf, jcrf, jfbkf, jocf, joidf, joslf, jsrf, jtsf*) require one ck clock cycle to be executed in both cases.

9.7.3.3 Unconditional Jump Relative

The unconditional jump relative instructions require that the uPC register be refilled so execution can continue at a new address. First, the effective address of the destination is calculated using the relative offset in the instruction. The address is loaded into the program counter (uPC). The execution continues at the new address.

The unconditional jump relative instruction *jmpr* requires one ck clock cycle to be executed.

9.7.3.4 Unconditional Jump Far

The execution of the unconditional jump far instructions require two steps.

The first step consists of loading the destination address in the 10-bit jump register jr1 or jr2 with the *load* and *cp* instructions.

The unconditional jump far instruction can then be executed. The two instruction steps can be consecutive. The jump instruction is not destructive for the jump register. The data that contain the jumps registers can be reused by another instruction.

The effective address of jump is calculated using the 10-bit absolute address previously loaded in the appropriate jump register. The suitable jump register is selected by an instruction operand among the two jump registers jr1 and jr2. The address is loaded into the program counter (uPC) and the execution can continue at a new address.

The unconditional jump far instruction *jmpf* requires one ck clock cycle to be executed.

9.7.3.5 Wait Table

The *wait* instruction uses a “wait table” to configure its behavior. The wait table is composed of five entries. Each entry contains:

- An enable flag (one bit). This flag is set by the wait instruction to select if the condition code specified in the entry is enabled.
- A condition code. (6-bit) This code specifies the condition to be tested.
- A destination address (10-bit). This address specifies the address of the Code RAM to which the program execution should jump if the wait condition is satisfied. Regardless of the addressing mode (DM or EM), the address stored in the wait table is always the physical address of the destination.

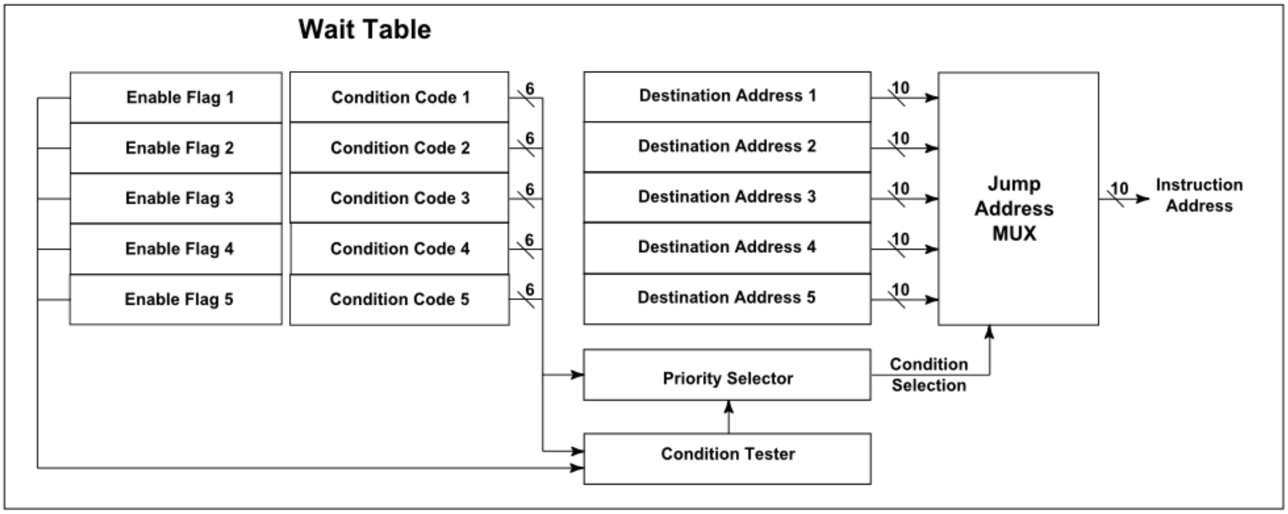


Figure 43. Wait Table Management Diagram

Before the wait instruction is called, the wait table has to be filled to configure the wait entries and obtain the desired behavior by means of the *cwef* and *cwer* instructions.

One instruction is required for each wait entry needing to be configured. The wait table is not reset after the execution of the *wait* instruction. If some of the entries are common between one *wait* instruction and the following one, the entry doesn’t need to be reconfigured but can be modified, added, or removed.

During the wait instruction execution the uPC register is not incremented until one of the condition defined in the wait table is satisfied. The effective destination address of the next instruction is calculated using the10-bit absolute address previously loaded in the appropriate jump register (if entry defined with a *cwef* instruction) or using the relative address (if entry defined with a *cwer* instruction). Then the address is loaded into the program counter (uPC) and the execution can continue at the new address. The wait table is not active until a *wait* instruction is executed again.

9.8 Load Instructions

The 33816 digital block offers a set of 9 instructions dedicated to loading microcore registers and Data RAM:

- The instruction *load* is used to load any registers (including ALU registers) from Data RAM.
- The instruction *store* is used to copy any registers (including ALU registers) to the Data RAM.
- The instruction *cp* is used to copy any registers (including ALU registers) to any registers (including ALU registers).
- The instructions *ldca* and *ldcd* are dedicated to the counter registers loading (eoc1, eoc2, eoc3, eoc4). The *ldca* instruction also controls of the pre-driver outputs.
- The instructions *ldirh* and *ldirh* are used for loading the ALU multiplication result register reg32.
- The instructions *ldjr1* and *ldjr2* are dedicated for loading the jump registers jr1 and jr2.

The load instructions (*load, store, cp, ldca, ldcd, ldir, ldirh, ldjr1, ldjr2*) require one ck clock cycle to be executed.

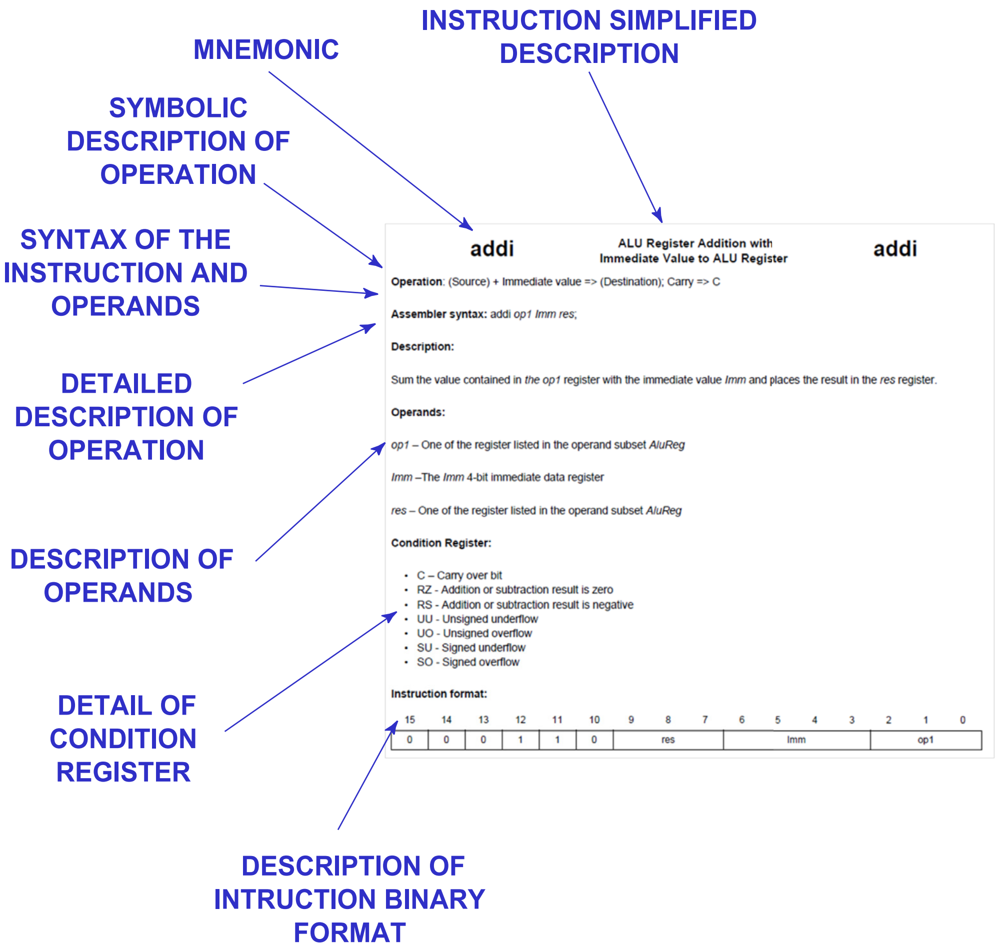
10 Instruction Glossary

10.1 Introduction

This section is a comprehensive reference to the MC33816 instruction set.

10.2 Glossary Information

The glossary contains an entry for each assembler mnemonic, in alphabetic order. Figure 44 is a representation of a glossary page.



addi ALU Register Addition with Immediate Value to ALU Register **addi**

Operation: (Source) + Immediate value => (Destination); Carry => C

Assembler syntax: addi *op1 Imm res*;

Description:

Sum the value contained in *the op1* register with the immediate value *Imm* and places the result in the *res* register.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Imm –The *Imm* 4-bit immediate data register

res – One of the register listed in the operand subset *AluReg*

Condition Register:

- C – Carry over bit
- RZ - Addition or subtraction result is zero
- RS - Addition or subtraction result is negative
- UU - Unsigned underflow
- UO - Unsigned overflow
- SU - Signed underflow
- SO - Signed overflow

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	res			Imm				op1		

Figure 44. Description of Glossary Page

10.3 Operand Subsets

This section details the pre-defined microcore register subsets used by instruction operand in Direct Addressing Mode (DM).

Table 225. Operand Subset Overview

Operand Label	Operand Subset Description
AluReg	Register designator for registers r0, r1, r2, r3, r5, r5, ir, mh, and ml.
AluGprIrReg	Register designator for registers r0, r1, r2, r3, r5, r5, and ir.
UcReg	Register designator for registers r0, r1, r2, r3, r5, r5, ir, mh, ml, ar (arith_reg), aux, jr1, jr2, cnt1, cnt2, cnt3, cnt4, eoc1, eoc1, eoc3, eoc4, flag, cr (ctrl_reg), sr (status_bits), spi_data, dac_sssc, dac_ossc, dac_ssoc, dac_osoc, dac4h4n, spi_add, irq (irq_status), and rxtx (ch_rxtx)
JpReg	Register designator for registers jr0 and jr1

10.3.1 *AluReg* Subset

Table 226. *AluReg* Subset Description

Register Label	Operand Binary Value
r0	000
r1	001
r2	010
r3	011
r4	100
ir	101
mh	110
ml	111

10.3.2 *AluGprIrReg* Subset

Table 227. *AluGpsIrReg* Subset Description

Register Label	Operand Binary Value
r0	000
r1	001
r2	010
r3	011
r4	100
ir	101

10.3.3 *UcReg* Subset

Table 228. *UcReg* Subset Description

Register Label	Operand Binary Value
r0	00000
r1	00001
r2	00010
r3	00011
r4	00100
ir	00101
mh	00110
ml	00111
ar (97)	01000
aux	01001
jr1	01010
jr2	01011
cnt1	01100
cnt2	01101
cnt3	01110
cnt4	01111
eoc1	10000
eoc2	10001
eoc3	10010
eoc4	10011
flag	10100
cr (97)	10101
sr (99)	10110
spi_data	10111
dac_sssc	11000
dac_ossc	11001
dac_ssoc	11010
dac_osoc	11011
dac4h4n	11100
spi_add	11101
irq (100)	11110

Table 228. *UcReg* Subset Description (continued)

Register Label	Operand Binary Value
rxtx (101)	11111

Notes

97. *ar* is the ALU arithmetic register arith_reg
98. *cr* is the control register ctrl_reg
99. *sr* is the status bits register status_bits
100. *irq* is the interrupt status register irq_status
101. *rxtx* is the other channel communication register ch_rxtx

10.3.4 *JpReg* Subset

Table 229. *JrReg* Subset Description

Register Label	Operand Binary Value
jr1	0
jr2	1

10.4 Glossary

This subsection contains an entry for each assembler mnemonic, in alphabetic order.

add — Two ALU registers addition to ALU register — add

Operation: (Source1) + (Source2) => (Destination); Carry => C

Assembler syntax: add *op1 op2 res*;

Description:

Sums the value contained in *the op1* register with the value contained in *op2* register and places the result in the *res* register.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

op2 – One of the register listed in the operand subset *AluReg*

res – One of the register listed in the operand subset *AluReg*

Condition Register:

- C – Carry over bit
- RZ - Addition or subtraction result is zero
- RS - Addition or subtraction result is negative
- UU - Unsigned underflow
- UO - Unsigned overflow
- SU - Signed underflow
- SO - Signed overflow

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	0	res			0	op2			op1		

addi ALU register addition with immediate value to ALU register addi

Operation: (Source) + Immediate value => (Destination); Carry => C

Assembler syntax: addi *op1 Imm res*;

Description:

Sums the value contained in *the op1* register with the immediate value *Imm* and places the result in the *res* register.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Imm –The *Imm* 4-bit immediate data register

res – One of the register listed in the operand subset *AluReg*

Condition Register:

- C – Carry over bit
- RZ - Addition or subtraction result is zero
- RS - Addition or subtraction result is negative
- UU - Unsigned underflow
- UO - Unsigned overflow
- SU - Signed underflow
- SO - Signed overflow

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	res			Imm				op1		

and AND-mask on ALU register with the immediate register to ALU register and

Operation: (Source)**?** Immediate register => (Source)

Assembler syntax: and *op1*;

Description:

Applies the AND-mask contained into the *Ir* register to the value contained in the *op1* register and places the result in the *op1* register. The initial data stored in the *op1* register is loss.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Ir –The ALU immediate register

Condition Register:

- MN - Mask result is 0x0000
- MM - Mask result is 0xFFFF

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	1	0	1	1	0	0	1	op1		

bias Enable high-side and low-side bias bias

Assembler syntax: bias *BiasTarget Ctrl*;

Description:

Enables/disables individually the high-side and low-side 33816 load bias structures.

This operation is successful only if the microcore has the right to drive the output related to the selected bias structure. The drive right is granted by setting the related bits in the Out_acc_ucX_chY (0x184, 0x185, 0x186, 0x187) configuration registers.

Operands:

BiasTarget – Operand that defines the bias structure(s) to be selected

Operand Label	Operand Description	Operand Binary Value
hs1	Select HS1 bias structure	0000
hs2	Select HS2 bias structure	0001
hs3	Select HS3 bias structure	0010
hs4	Select HS4 bias structure	0011
hs5	Select HS5 bias structure	0100
ls1	Select LS1 bias structure	0101
ls2	Select LS2 bias structure	0110
ls3	Select LS3 bias structure	0111
ls4	Select LS4 bias structure	1000
ls5	Select LS5 bias structure	1001
hs2s	Select HS2 strong bias structure	1010
hs4s	Select HS4 strong bias structure	1011
ls6	Select LS6 bias structure	1100
all	Select all high-side and low-side pre-driver bias structures including strong bias structures	1101
hs	Select all high-side pre-driver bias structures including strong bias structures	1110
ls	Select all low-side pre-driver bias structures	1111

Ctrl – Operand that define the bias structure(s) state to be applied

Operand Label	Operand Description	Operand Binary Value
off	Bias structure disable	0
on	Bias structure enable	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	1	0	0	0	1	Ctrl	BiasTarget			

chth Change V_{DS} and V_{SRC} threshold chth

Assembler syntax: chth *SelFbk ThLevel*;

Description:

Changes the thresholds for the selected V_{DS} and V_{SRC} feedback comparator.

This operation is successful only if the microcore has the right to drive the output related to selected threshold.
The configuration of the high-side pre-driver Vsrc thresholds is also impacted by the bootstrap initialization mode.

Operands:

SelFbk – Operand that defines the threshold comparator to be selected

Operand Label	Operand Description	Operand Binary Value
hs1v	HS1 V_{DS} feedback	0000
hs1s	HS1 V_{SRC} feedback	0001
hs2v	HS2 V_{DS} feedback	0010
hs2s	HS2 V_{SRC} feedback	0011
hs3v	HS3 V_{DS} feedback	0100
hs3s	HS3 V_{SRC} feedback	0101
hs4v	HS4 V_{DS} feedback	0110
hs4s	HS4 V_{SRC} feedback	0111
hs5v	HS5 V_{DS} feedback	1000
hs5s	HS5 V_{SRC} feedback	1001
ls1v	HS1 V_{DS} feedback	1010
ls2v	HS2 V_{DS} feedback	1011
ls3v	HS3 V_{DS} feedback	1100
ls4v	HS4 V_{DS} feedback	1101
ls5v	HS5 V_{DS} feedback	1110
ls6v	HS6 V_{DS} feedback	1111

ThLevel – Operand that defines threshold level to be applied

Operand Label	Operand Description	Operand Binary Value
lv1	First level	000
lv2	Second level	001
lv3	Third level	010
lv4	Fourth level	011
lv5	Fifth level	100
lv6	Sixth level	101
lv7	Seventh level	110
lv8	Height level	111

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	1	1	1	SelFbk				ThLevel		

cp

Copy source register data in destination register

cp

Assembler syntax: cp *op1 op2;*

Description:

Copies the value from the source register *op1* into the destination register *op2*.

Operands:

op1 – One of the register listed in the operand subset *UcReg*

op2 – One of the register listed in the operand subset *UcReg*

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	op1					op2					0	0	0

cwef Create wait table entry far cwef

Assembler syntax: cwef *op1 Cond Entry ;*

Description:

Initializes or changes a row in the wait table used by the *wait* instruction

The wait table is a five-row/two-column table:

- The first column contains the wait conditions.
- The second column contains the jump register name op1 that contains the absolute destination addresses.

Up to 5 conditions may be checked at the same time.

When the condition Cond is satisfied and the entry is enabled, the execution continues at the corresponding destination jump address.

Operands:

op1 – One of the register listed in the operand subset *JpReg*

Cond – Operand that defines the condition to be satisfied to enable the jump far

Operand Label	Operand Description	Operand Binary Value
_f0	Flag 0 low	000000
_f1	Flag 1 low	000001
_f2	Flag 2 low	000010
_f3	Flag 3 low	000011
_f4	Flag 4 low	000100
_f5	Flag 5 low	000101
_f6	Flag 6 low	000110
_f7	Flag 7 low	000111
_f8	Flag 8 low	001000
_f9	Flag 9 low	001001
_f10	Flag 10 low	001010
_f11	Flag 11 low	001011
_f12	Flag 12 low	001100
_f13	Flag 13 low	001101
_f14	Flag 14 low	001110
_f15	Flag 15 low	001111
f0	Flag 0 high	010000
f1	Flag 1 high	010001
f2	Flag 2 high	010010

Operand Label	Operand Description	Operand Binary Value
f3	Flag 3 high	010011
f4	Flag 4 high	010100
f5	Flag 5 high	010101
f6	Flag 6 high	010110
f7	Flag 7 high	010111
f8	Flag 8 high	011000
f9	Flag 9 high	011001
f10	Flag 10 high	011010
f11	Flag 11 high	011011
f12	Flag 12 high	011100
f13	Flag 13 high	011101
f14	Flag 14 high	011110
f15	Flag 15 high	011111
tc1	Terminal count 1	100000
tc2	Terminal count 2	100001
tc3	Terminal count 3	100010
tc4	Terminal count 4	100011
_start	Start low	100100
start	Start high	100101
_sc1v	Shortcut1 VDS feedback low	100110
_sc2v	Shortcut2 VDS feedback low	100111
_sc3v	Shortcut3 VDS feedback low	101000
_sc1s	Shortcut1 source feedback low	101001
_sc2s	Shortcut2 source feedback low	101010
_sc3s	Shortcut3 source feedback low	101011
sc1v	Shortcut1 VDS feedback high	101100
sc2v	Shortcut2 VDS feedback high	101101
sc3v	Shortcut3 VDS feedback high	101110
opd	Instruction request to ALU executed	101111
vb	Boost voltage high	110000
_vb	Boost voltage low	110001
cur1	Current feedback 1 high	110010
cur2	Current feedback 2 high	110011
cur3	Current feedback 3 high	110100
cur4l	Current feedback 4l high	110101
cur4h	Current feedback 4h high	110110
cur4n	Current feedback 4n high	110111
_cur1	Current feedback 1 low	111000
_cur2	Current feedback 2 low	111001

Operand Label	Operand Description	Operand Binary Value
_cur3	Current feedback 3 low	111010
_cur4l	Current feedback 4l low	111011
_cur4h	Current feedback 4h low	111100
_cur4n	Current feedback 4n low	111101
ocur	Own current feedback high	111110
_ocur	Own current feedback low	111111

Entry – Operand that defines the wait table row number

Operand Label	Operand Description	Operand Binary Value
row1	Wait table row 1	000
row2	Wait table row 2	001
row3	Wait table row 3	010
row4	Wait table row 4	011
row5	Wait table row 5	100

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	op1	Entry			Cond					

cwer Create wait table entry relative **cwer**

Assembler syntax: cwer *Dest Cond Entry ;*

Description:

Initializes or changes a row in the wait table used by the *wait* instruction

The wait table is a five-row/two-column table:

- The first column contains the wait conditions
- The second column contains the destination jump addresses

Up to five conditions may be checked at the same time.

When the condition Cond is satisfied and the entry is enabled, the execution continues at the correspondent destination jump address.

The jump is relative to the instruction Code RAM location. The destination address is the actual instruction Code RAM location added to the *Dest* operand value. This 5-bit value is a two's complemented number. The MSB is the sign. So *Dest* operand value is in the range of {-16, 15}.

Operands:

Dest – Operand that defines the 5-bit relative destination address in the range of {-16, 15}

Cond – Operand that defines the condition to be satisfied to enable the jump far

Operand Label	Operand Description	Operand Binary Value
_f0	Flag 0 low	000000
_f1	Flag 1 low	000001
_f2	Flag 2 low	000010
_f3	Flag 3 low	000011
_f4	Flag 4 low	000100
_f5	Flag 5 low	000101
_f6	Flag 6 low	000110
_f7	Flag 7 low	000111
_f8	Flag 8 low	001000
_f9	Flag 9 low	001001
_f10	Flag 10 low	001010
_f11	Flag 11 low	001011
_f12	Flag 12 low	001100
_f13	Flag 13 low	001101
_f14	Flag 14 low	001110
_f15	Flag 15 low	001111

Operand Label	Operand Description	Operand Binary Value
f0	Flag 0 high	010000
f1	Flag 1 high	010001
f2	Flag 2 high	010010
f3	Flag 3 high	010011
f4	Flag 4 high	010100
f5	Flag 5 high	010101
f6	Flag 6 high	010110
f7	Flag 7 high	010111
f8	Flag 8 high	011000
f9	Flag 9 high	011001
f10	Flag 10 high	011010
f11	Flag 11 high	011011
f12	Flag 12 high	011100
f13	Flag 13 high	011101
f14	Flag 14 high	011110
f15	Flag 15 high	011111
tc1	Terminal count 1	100000
tc2	Terminal count 2	100001
tc3	Terminal count 3	100010
tc4	Terminal count 4	100011
_start	Start low	100100
start	Start high	100101
_sc1v	Shortcut1 VDS feedback low	100110
_sc2v	Shortcut2 VDS feedback low	100111
_sc3v	Shortcut3 VDS feedback low	101000
_sc1s	Shortcut1 source feedback low	101001
_sc2s	Shortcut2 source feedback low	101010
_sc3s	Shortcut3 source feedback low	101011
sc1v	Shortcut1 VDS feedback high	101100
sc2v	Shortcut2 VDS feedback high	101101
sc3v	Shortcut3 VDS feedback high	101110
opd	Instruction request to ALU executed	101111
vb	Boost voltage high	110000
_vb	Boost voltage low	110001
cur1	Current feedback 1 high	110010
cur2	Current feedback 2 high	110011
cur3	Current feedback 3 high	110100
cur4l	Current feedback 4l high	110101
cur4h	Current feedback 4h high	110110

Operand Label	Operand Description	Operand Binary Value
cur4n	Current feedback 4n high	110111
_cur1	Current feedback 1 low	111000
_cur2	Current feedback 2 low	111001
_cur3	Current feedback 3 low	111010
_cur4l	Current feedback 4l low	111011
_cur4h	Current feedback 4h low	111100
_cur4n	Current feedback 4n low	111101
ocur	Own current feedback high	111110
_ocur	Own current feedback low	111111

Entry – Operand that defines the wait table row number

Operand Label	Operand Description	Operand Binary Value
row1	Wait table row 1	000
row2	Wait table row 2	001
row3	Wait table row 3	010
row4	Wait table row 4	011
row5	Wait table row 5	100

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	Dest					Entry			Cond					

dfcsct Define current shortcut dfcsct

Assembler syntax: dfcsct *ShrtCur ;*

Description:

Defines the shortcut for the current feedback.

This shortcut defines the connection between the physical current feedback input of the microcore and the current measurement block.

At reset the default shortcut setting is the following:

Shortcut	Uc0Ch1	Uc1Ch1	Uc0Ch2	Uc1Ch2
ShrtCur	dac1	dac2	dac3	dac4l

Operands:

ShrtCur – Operand that defines to which current measurement block is dedicated the shortcut.

Operand Label	Operand Description	Operand Binary Value
dac1	DAC1 is selected as current shortcut	00
dac2	DAC2 is selected as current shortcut	01
dac3	DAC3 is selected as current shortcut	10
dac4l	DAC4l is selected as current shortcut	11

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	0	1	1	1	1	0	ShrtCur	

dfsct Define pre-driver output shortcuts **dfsct**

Assembler syntax: dfsct *Shrt1 Shrt2 Shrt3;*

Description:

Defines three shortcuts applied to three pre-drivers output among the set of all the low-side and high-side pre-drivers.

The shortcuts table defines the connection between the physical outputs of the microcore and the external outputs pin (G_HSx and G_LSx) driving the MOSFETs.

At reset the default shortcut setting is the following:

Shortcut	Channel 1		Channel 2	
	microcore 0	microcore 1	microcore 0	microcore 1
Shrt1	hs1	hs2	hs3	hs4
Shrt2	ls1	ls2	ls3	ls4
Shrt3	ls5	ls6	ls7	hs5

Operands:

Shrt1, Shrt2, and Shrt3 – Operands that define to which pre-driver is dedicated the shortcut.

Operand Label	Operand Description	Operand Binary Value
hs1	High-side pre-driver 1	0000
hs2	High-side pre-driver 2	0001
hs3	High-side pre-driver 3	0010
hs4	High-side pre-driver 4	0011
hs5	High-side pre-driver 5	0100
ls1	Low-side pre-driver 1	0101
ls2	Low-side pre-driver 2	0110
ls3	Low-side pre-driver 3	0111
ls4	Low-side pre-driver 4	1000
ls5	Low-side pre-driver 5	1001
ls6	Low-side pre-driver 6	1010
ls7	Low-side pre-driver 7	1011
undef	Undefined shortcut	1100

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	Shrt1				Shrt2				Shrt3				1	1

endiag Enable automatic diagnosis endiag

Assembler syntax: endiag *Sel Diag*;

Description:

Enables or disables the automatic diagnosis for a single output and the related interrupt procedure for error handling.

This operation is successful only if the microcore has the right to drive the related outputs. The drive right is granted by setting the related bits in the Out_acc_ucX_chY (0x184, 0x185, 0x186, 0x187) configuration registers.

At reset the automatic diagnosis is disabled.

Operands:

Sel – Operand that defines the monitored pre-driver and V_{DS} or V_{SRC} feedback.

Operand Label	Operand Description	Operand Binary Value
hs1v	High-side pre-driver 1 vds feedback	0000
hs1s	High-side pre-driver 1 src feedback	0001
hs2v	High-side pre-driver 2 vds feedback	0010
hs2s	High-side pre-driver 2 src feedback	0011
hs3v	High-side pre-driver 3 vds feedback	0100
hs3s	High-side pre-driver 3 src feedback	0101
hs4v	High-side pre-driver 4 vds feedback	0110
hs4s	High-side pre-driver 4 src feedback	0111
hs5v	High-side pre-driver 5 vds feedback	1000
hs5s	High-side pre-driver 5 src feedback	1001
ls1v	Low-side pre-driver 1 vds feedback	1010
ls2v	Low-side pre-driver 2 vds feedback	1011
ls3v	Low-side pre-driver 3 vds feedback	1100
ls4v	Low-side pre-driver 4 vds feedback	1101
ls5v	Low-side pre-driver 5 vds feedback	1110
ls6v	Low-side pre-driver 6 vds feedback	1111

Diag – Operand that defines the diagnosis status

Operand Label	Operand Description	Operand Binary Value
diagoff	Automatic diagnosis disable	0
diagon	Automatic diagnosis enable	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	0	1	1	0	Sel				Diag

endiaga Enable all automatic diagnosis endiaga

Assembler syntax: endiaga *Diag*;

Description:

Enables or disables the automatic diagnosis for all the pre-drivers output that the microcore is configured to drive. If automatic diagnosis condition is satisfied, the related interrupt procedure for error handling is triggered.

The operation is successful only if the microcore has the right to drive the related outputs. The drive right is granted by setting the related bits in the Out_acc_ucX_chY (0x184, 0x185, 0x186, 0x187) configuration registers.

At reset the automatic diagnosis is disabled.

Operands:

Diag – Operand that defines the diagnosis status

Operand Label	Operand Description	Operand Binary Value
diagoff	Automatic diagnosis disable	0
diagon	Automatic diagnosis enable	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	Diag

endiags Enable automatic diagnosis shortcuts endiags

Assembler syntax: endiags *Diag_sh1_vds Diag_sh1_src Diag_sh2_vds Diag_sh3_vds*;

Description:

Enables or disables the automatic for the outputs selected via shortcuts

Four events can be monitored in parallel:

- the drain-source voltage on shortcut1 output (*Diag_sh1_vds*)
- the source voltage on shortcut1 output (*Diag_sh1_src*)
- the drain-source voltage on shortcut2 output (*Diag_sh2_vds*)
- the drain-source voltage on shortcut3 output (*Diag_sh3_vds*)

If automatic diagnosis condition is satisfied, the related interrupt procedure for error handling is triggered.

The shortcuts are defined with the dfsct instruction.

The operation is successful only if the microcore has the right to drive the related outputs. The drive right is granted by setting the related bits in the Out_acc_ucX_chY (0x184, 0x185, 0x186, 0x187) configuration registers.

At reset the automatic diagnosis are disabled.

Operands:

Diag _sh1_vds, Diag_sh2_vds and *Diag_sh3_vds* – Operands corresponding to the shortcuts related to V_{DS} to be monitored.

Operand Label	Operand Description	Operand Binary Value
keep	No changes, maintains the previous setting	00
NA	Not applicable	01
off	Automatic diagnosis disabled	10
on	Automatic diagnosis enabled	11

Diag _sh1_src – Operand corresponding to the shortcuts related to V_{SRC} to be monitored.

Operand Label	Operand Description	Operand Binary Value
keep	No changes, maintains the previous setting	00
NA	Not applicable	01
off	Automatic diagnosis disabled	10
on	Automatic diagnosis enabled	11

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	1	1	0	Diag _sh1_vds		Diag _sh1_src		Diag_sh2_vds		Diag_sh3_vds	

iconf Interrupt configuration iconf

Assembler syntax: iconf *Conf*;

Description:

Configures the microcore to be enabled by the interrupt return request.

The automatic interrupt return request is issued from, according to the iret_en bit state of the Driver_config register (0x1C5):

- Re-enabling the drivers in case the disabled drivers interrupt.
- Reading or writing the Driver_status register (0x1D2) in case of automatic diagnosis interrupt. This register must be configured such as to be ‘reset at read’.

The reset value is *none*.

Operands:

Conf – Operand that defines interrupt behaviors

Operand Label	Operand Description	Operand Binary Value
none	The microcore ignores all automatic interrupt return request	00
NA	Not applicable	01
continue	When an interrupt return request is received, the code execution continues from where it was interrupted	10
restart	When an interrupt return request is received, the code execution restarts from the entry point	11

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	1	0	0	0	0	0	0	1	Conf	

iret Return from interrupt **iret**

Assembler syntax: iret *Type Rst*;

Description:

Ends the interrupt routine and clears the microcore Interrupt_status register (0x1D4).

Operands:

Type – Operand that defines how the program counter (uPC) is handled returning from the interrupt routine

Operand Label	Operand Description	Operand Binary Value
continue	The execution is resumed at the address stored in the 10 LSBs of the Interrupt_status register (0x1D4) (iret microcore register)	0
restart	The execution is resumed at the address stored in the Ucx_entry_point registers (0x10A, 0x10B, 0x12A, 0x12B)	1

Rst – Operand that defines if the pending interrupts queue is clear when the *iret* instruction is executed

Operand Label	Operand Description	Operand Binary Value
_rst	The pending interrupts queue is not cleared	0
rst	The pending interrupts queue is cleared	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	1	0	0	1	1	0	Type	Rst

jarf Jump far on arithmetic condition jarf

Assembler syntax: jarf *op1 BitSel*;

Description:

Configures the jump to absolute location on arithmetic condition.

If the condition defined by the *BitSel* operand is satisfied, the program counter (uPC) is handled such as the next executed instruction is located into the destination address contained in one of the jump registers.

The destination address defined by the *op1* register is any of the absolute Code RAM location.

Operands:

op1 – One of the register listed in the operand subset *JpReg*

BitSel – Operand that defines the arithmetic condition that trigs the jump. The arithmetic conditions are stored into the ALU condition register

Operand Label	Operand Description	Operand Binary Value
opd	OD -Operation complete	0000
ovs	SO - Overflow with signed operands	0001
uns	SU - Underflow with signed operands	0010
ovu	UO - Overflow with unsigned operands	0011
unu	UU - Underflow with unsigned operands	0100
sgn	CS - Sign of result	0101
zero	RZ - Result is zero	0110
mloss	ML - Multiply precision loss	0111
mover	MO - Multiply overflow	1000
all1	MM - Result of mask operation is 0xFFFF	1001
all0	MN - Result of mask operation is 0x0000	1010
aritl	A0	1011
arith	A1	1100
carry	C - Carry	1101
conv	CS - Conversion sign	1110
csh	SB - Carry on shift operation	1111

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	0	1	BitSel				op1	0	1	0	1

jarr Jump relative on arithmetic condition jarr

Assembler syntax: jarr *Dest BitSel*;

Description:

Configures jump to relative location on arithmetic condition.

If the condition defined by the *BitSel* operand is satisfied, the program counter (uPC) is handled such as the next executed instruction is relative destination address.

The jump is relative to the instruction Code RAM location. The destination address is the actual instruction Code RAM location added to the Dest operand value. This 5-bit value is a two’s complemented number. The MSB is the sign. So *Dest* operand value is in the range of {-16, 15}.

Operands:

Dest – Operand that defines the 5-bit relative destination address in the range of {-16, 15}.

BitSel – Operand that defines the arithmetic condition that trigs the jump. The arithmetic conditions are stored into the ALU condition register

Operand Label	Operand Description	Operand Binary Value
opd	OD -Operation complete	0000
ovs	SO - Overflow with signed operands	0001
uns	SU - Underflow with signed operands	0010
ovu	UO - Overflow with unsigned operands	0011
unu	UU - Underflow with unsigned operands	0100
sgn	CS - Sign of result	0101
zero	RZ - Result is zero	0110
mloss	ML - Multiply precision loss	0111
mover	MO - Multiply overflow	1000
all1	MM - Result of mask operation is 0xFFFF	1001
all0	MN - Result of mask operation is 0x0000	1010
aritl	A0	1011
arith	A1	1100
carry	C - Carry	1101
conv	CS - Conversion sign	1110
csh	SB - Carry on shift operation	1111

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	1	0	BitSel				Dest				

jcrf Jump far on control register condition jcrf

Assembler syntax: jcrf *op1 CrSel Pol*;

Description:

Configures the jump to absolute location on control register condition.

If the condition defined by the *CrSel* operand is satisfied according to the polarity *Pol,* the program counter (uPC) is handled such as the next executed instruction is located into the destination address contained in one of the jump registers.

The destination address defined by the *op1* register is any of the absolute Code RAM location.

Operands:

op1 – One of the register listed in the operand subset *JpReg*

CrSel – Operand that defines the control register condition (Ctrl_reg_uc0 and Ctrl_reg_uc1 registers (0x101, 0x102, 0x121, 0x122)) that trigs the jump

Operand Label	Operand Description	Operand Binary Value
b0	Control register bit 0 (LSB)	0000
b1	Control register bit 1	0001
b2	Control register bit 2	0010
b3	Control register bit 3	0011
b4	Control register bit 4	0100
b5	Control register bit 5	0101
b6	Control register bit 6	0110
b7	Control register bit 7	0111
b8	Control register bit 8	1000
b9	Control register bit 9	1001
b10	Control register bit 10	1010
b11	Control register bit 11	1011
b12	Control register bit 12	1100
b13	Control register bit 13	1101
b14	Control register bit 14	1110
b15	Control register bit 15 (MSB)	1111

Pol – Operand that defines the active polarity for the selected bit

Operand Label	Operand Description	Operand Binary Value
low	Active condition if the selected bit is ‘0’	0
high	Active condition if the selected bit is ‘1’	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	1	Pol	CrSel				op1	0	1	0	0

jcrr Jump relative on control register condition jcrr

Assembler syntax: jcrr *Dest CrSel Pol*;

Description:

Configures the jump to relative location on control register condition.

If the condition defined by the *CrSel* operand is satisfied according to the polarity *Pol*, the program counter (uPC) is handled such as the next executed instruction is relative destination address

The jump is relative to the instruction Code RAM location. The destination address is the actual instruction Code RAM location added to the Dest operand value. This 5-bit value is a two's complemented number. The MSB is the sign. So *Dest* operand value is in the range of {-16, 15}.

Operands:

Dest – Operand that defines the 5-bit relative destination address in the range of {-16, 15}.

CrSel – Operand that defines the control register condition (Ctrl_reg_uc0 and Ctrl_reg_uc1 registers (0x101, 0x102, 0x121, 0x122)) that trigs the jump.

Operand Label	Operand Description	Operand Binary Value
b0	Control register bit 0 (LSB)	0000
b1	Control register bit 1	0001
b2	Control register bit 2	0010
b3	Control register bit 3	0011
b4	Control register bit 4	0100
b5	Control register bit 5	0101
b6	Control register bit 6	0110
b7	Control register bit 7	0111
b8	Control register bit 8	1000
b9	Control register bit 9	1001
b10	Control register bit 10	1010
b11	Control register bit 11	1011
b12	Control register bit 12	1100
b13	Control register bit 13	1101
b14	Control register bit 14	1110
b15	Control register bit 15 (MSB)	1111

Pol – Operand that defines the active polarity for the selected bit

Operand Label	Operand Description	Operand Binary Value
low	Active condition if the selected bit is ‘0’	0
high	Active condition if the selected bit is ‘1’	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	0	Pol	CrSel				Dest				

jfbkf Jump far on feedback condition **jfbkf**

Assembler syntax: jfbkf *op1 SelFbk Pol*;

Description:

Configures the jump to absolute location on feedback condition.

If the condition defined by the *SelFbk* operand is satisfied according to the polarity *Pol,* the program counter (uPC) is handled such as the next executed instruction is located into the destination address contained in one of the jump registers.

The destination address defined by the *op1* register is any of the absolute Code RAM location.

Operands:

op1 – One of the register listed in the operand subset *JpReg*

SelFbk – Operand that defines the feedback signal condition

Operand Label	Operand Description	Operand Binary Value
hs1v	High-side pre-driver 1 V_{DS} feedback	0000
hs1s	High-side pre-driver 1 V_{SRC} feedback	0001
hs2v	High-side pre-driver 2 V_{DS} feedback	0010
hs2s	High-side pre-driver 2 V_{SRC} feedback	0011
hs3v	High-side pre-driver 3 V_{DS} feedback	0100
hs3s	High-side pre-driver 3 V_{SRC} feedback	0101
hs4v	High-side pre-driver 4 V_{DS} feedback	0110
hs4s	High-side pre-driver 4 V_{SRC} feedback	0111
hs5v	High-side pre-driver 5 V_{DS} feedback	1000
hs5s	High-side pre-driver 5 V_{SRC} feedback	1001
ls1v	Low-side pre-driver 1 V_{DS} feedback	1010
ls2v	Low-side pre-driver 2 V_{DS} feedback	1011
ls3v	Low-side pre-driver 3 V_{DS} feedback	1100
ls4v	Low-side pre-driver 4 V_{DS} feedback	1101
ls5v	Low-side pre-driver 5 V_{DS} feedback	1110
ls6v	Low-side pre-driver 6 V_{DS} feedback	1111

Pol – Operand that defines the active polarity for the selected bit

Operand Label	Operand Description	Operand Binary Value
low	Active condition if the selected bit is ‘0’	0
high	Active condition if the selected bit is ‘1’	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	0	SelFbk				Pol	op1	0	1	0	0

jfbkr Jump relative on feedback condition jfbkr

Assembler syntax: jfbkr *Dest SelFbk Pol*;

Description:

Configures the jump to relative location on feedback condition.

If the condition defined by the *SelFbk* operand is satisfied according to the polarity *Pol,* the program counter (uPC) is handled such as the next executed instruction is relative destination address.

The jump is relative to the instruction Code RAM location. The destination address is the actual instruction Code RAM location added to the Dest operand value. This 5-bit value is a two's complemented number. The MSB is the sign. So *Dest* operand value is in the range of {-16, 15}.

Operands:

Dest – Operand that defines the 5-bit relative destination address in the range of {-16, 15}.

SelFbk – Operand that defines the feedback signal condition

Operand Label	Operand Description	Operand Binary Value
hs1v	High-side pre-driver 1 V_{DS} feedback	0000
hs1s	High-side pre-driver 1 V_{SRC} feedback	0001
hs2v	High-side pre-driver 2 V_{DS} feedback	0010
hs2s	High-side pre-driver 2 V_{SRC} feedback	0011
hs3v	High-side pre-driver 3 V_{DS} feedback	0100
hs3s	High-side pre-driver 3 V_{SRC} feedback	0101
hs4v	High-side pre-driver 4 V_{DS} feedback	0110
hs4s	High-side pre-driver 4 V_{SRC} feedback	0111
hs5v	High-side pre-driver 5 V_{DS} feedback	1000
hs5s	High-side pre-driver 5 V_{SRC} feedback	1001
ls1v	Low-side pre-driver 1 V_{DS} feedback	1010
ls2v	Low-side pre-driver 2 V_{DS} feedback	1011
ls3v	Low-side pre-driver 3 V_{DS} feedback	1100
ls4v	Low-side pre-driver 4 V_{DS} feedback	1101
ls5v	Low-side pre-driver 5 V_{DS} feedback	1110
ls6v	Low-side pre-driver 6 V_{DS} feedback	1111

Pol – Operand that defines the active polarity for the selected bit

Operand Label	Operand Description	Operand Binary Value
low	Active condition if the selected bit is ‘0’	0
high	Active condition if the selected bit is ‘1’	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	SelFbk				Pol	Dest				

jmpf Unconditional jump far jmpf

Assembler syntax: jmpf *op1*;

Description:

Configures the unconditional jump.

The destination address defined in one of the jump registers defined by the operand *op1*. The destination address is any of the absolute Code RAM location.

Operands:

op1 – One of the register listed in the operand subset *JpReg*

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	0	0	1	1	0	1	0	op1	1	0	1

jmpr Unconditional jump relative jmpr

Assembler syntax: jmpr *Dest SelFbk Pol*;

Description:

Configures the unconditional jump to relative location.

The jump is relative to the instruction Code RAM location. The destination address is the actual instruction Code RAM location added to the Dest operand value. This 5-bit value is a two's complemented number. The MSB is the sign. So *Dest* operand value is in the range of {-16, 15}.

Operands:

Dest – Operand that defines the 5-bit relative destination address in the range of {-16, 15}.

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	1	1	0	0	0	Dest				

jocf Jump far on condition jocf

Assembler syntax: jocf *op1 Cond*;

Description:

Configures the jump to absolute location on condition.

If the condition defined by the *Cond* operand is satisfied, the program counter (uPC) is handled such as the next executed instruction is located into the destination address contained in one of the jump registers.

The destination address defined by the *op1* register is any of the absolute Code RAM location.

Operands:

op1 – One of the register listed in the operand subset *JpReg*

Cond – Operand that defines the condition to be satisfied to enable the jump far

Operand Label	Operand Description	Operand Binary Value
_f0	Flag 0 low	000000
_f1	Flag 1 low	000001
_f2	Flag 2 low	000010
_f3	Flag 3 low	000011
_f4	Flag 4 low	000100
_f5	Flag 5 low	000101
_f6	Flag 6 low	000110
_f7	Flag 7 low	000111
_f8	Flag 8 low	001000
_f9	Flag 9 low	001001
_f10	Flag 10 low	001010
_f11	Flag 11 low	001011
_f12	Flag 12 low	001100
_f13	Flag 13 low	001101
_f14	Flag 14 low	001110
_f15	Flag 15 low	001111
f0	Flag 0 high	010000
f1	Flag 1 high	010001
f2	Flag 2 high	010010
f3	Flag 3 high	010011
f4	Flag 4 high	010100
f5	Flag 5 high	010101

Operand Label	Operand Description	Operand Binary Value
f6	Flag 6 high	010110
f7	Flag 7 high	010111
f8	Flag 8 high	011000
f9	Flag 9 high	011001
f10	Flag 10 high	011010
f11	Flag 11 high	011011
f12	Flag 12 high	011100
f13	Flag 13 high	011101
f14	Flag 14 high	011110
f15	Flag 15 high	011111
tc1	Terminal count 1	100000
tc2	Terminal count 2	100001
tc3	Terminal count 3	100010
tc4	Terminal count 4	100011
_start	Start low	100100
start	Start high	100101
_sc1v	Shortcut1 VDS feedback low	100110
_sc2v	Shortcut2 VDS feedback low	100111
_sc3v	Shortcut3 VDS feedback low	101000
_sc1s	Shortcut1 source feedback low	101001
_sc2s	Shortcut2 source feedback low	101010
_sc3s	Shortcut3 source feedback low	101011
sc1v	Shortcut1 VDS feedback high	101100
sc2v	Shortcut2 VDS feedback high	101101
sc3v	Shortcut3 VDS feedback high	101110
opd	Instruction request to ALU executed	101111
vb	Boost voltage high	110000
_vb	Boost voltage low	110001
cur1	Current feedback 1 high	110010
cur2	Current feedback 2 high	110011
cur3	Current feedback 3 high	110100
cur4l	Current feedback 4l high	110101
cur4h	Current feedback 4h high	110110
cur4n	Current feedback 4n high	110111
_cur1	Current feedback 1 low	111000
_cur2	Current feedback 2 low	111001
_cur3	Current feedback 3 low	111010
_cur4l	Current feedback 4l low	111011
_cur4h	Current feedback 4h low	111100

Operand Label	Operand Description	Operand Binary Value
_cur4n	Current feedback 4n low	111101
ocur	Own current feedback high	111110
_ocur	Own current feedback low	111111

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	Cond						op1	0	0	0	0

jocr Jump relative on condition **jocr**

Assembler syntax: jocr *Dest Cond*;

Description:

Configures the jump to relative location on condition.

If the condition defined by the *Cond* operand is satisfied, the program counter (uPC) is handled such as the next executed instruction is relative destination address.

The jump is relative to the instruction Code RAM location. The destination address is the actual instruction Code RAM location added to the Dest operand value. This 5-bit value is a two's complemented number. The MSB is the sign. So *Dest* operand value is in the range of {-16, 15}.

Operands:

Dest – Operand that defines the 5-bit relative destination address in the range of {-16, 15}.

Cond – Operand that defines the condition to be satisfied to enable the relative jump

Operand Label	Operand Description	Operand Binary Value
_f0	Flag 0 low	000000
_f1	Flag 1 low	000001
_f2	Flag 2 low	000010
_f3	Flag 3 low	000011
_f4	Flag 4 low	000100
_f5	Flag 5 low	000101
_f6	Flag 6 low	000110
_f7	Flag 7 low	000111
_f8	Flag 8 low	001000
_f9	Flag 9 low	001001
_f10	Flag 10 low	001010
_f11	Flag 11 low	001011
_f12	Flag 12 low	001100
_f13	Flag 13 low	001101
_f14	Flag 14 low	001110
_f15	Flag 15 low	001111
f0	Flag 0 high	010000
f1	Flag 1 high	010001
f2	Flag 2 high	010010
f3	Flag 3 high	010011
f4	Flag 4 high	010100

Operand Label	Operand Description	Operand Binary Value
f5	Flag 5 high	010101
f6	Flag 6 high	010110
f7	Flag 7 high	010111
f8	Flag 8 high	011000
f9	Flag 9 high	011001
f10	Flag 10 high	011010
f11	Flag 11 high	011011
f12	Flag 12 high	011100
f13	Flag 13 high	011101
f14	Flag 14 high	011110
f15	Flag 15 high	011111
tc1	Terminal count 1	100000
tc2	Terminal count 2	100001
tc3	Terminal count 3	100010
tc4	Terminal count 4	100011
_start	Start low	100100
start	Start high	100101
_sc1v	Shortcut1 VDS feedback low	100110
_sc2v	Shortcut2 VDS feedback low	100111
_sc3v	Shortcut3 VDS feedback low	101000
_sc1s	Shortcut1 source feedback low	101001
_sc2s	Shortcut2 source feedback low	101010
_sc3s	Shortcut3 source feedback low	101011
sc1v	Shortcut1 VDS feedback high	101100
sc2v	Shortcut2 VDS feedback high	101101
sc3v	Shortcut3 VDS feedback high	101110
opd	Instruction request to ALU executed	101111
vb	Boost voltage high	110000
_vb	Boost voltage low	110001
cur1	Current feedback 1 high	110010
cur2	Current feedback 2 high	110011
cur3	Current feedback 3 high	110100
cur4l	Current feedback 4l high	110101
cur4h	Current feedback 4h high	110110
cur4n	Current feedback 4n high	110111
_cur1	Current feedback 1 low	111000
_cur2	Current feedback 2 low	111001
_cur3	Current feedback 3 low	111010
_cur4l	Current feedback 4l low	111011

Operand Label	Operand Description	Operand Binary Value
_cur4h	Current feedback 4h low	111100
_cur4n	Current feedback 4n low	111101
ocur	Own current feedback high	111110
_ocur	Own current feedback low	111111

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	Cond						Dest				

joidf Jump far on microcore condition **joidf**

Assembler syntax: joidf *op1 UcSel*;

Description:

Configures the jump to absolute location on microcore identifier condition.

If the condition defined by the *UcSel* operand is satisfied, the program counter (uPC) is handled such as the next executed instruction is located into the destination address contained in one of the jump registers.

The destination address defined by the *op1* register is any of the absolute Code RAM location.

Operands:

op1 – One of the register listed in the operand subset *JpReg*

UcSel – Operand that defines the microcore identifier condition

Operand Label	Operand Description	Operand Binary Value
seq0	The microcore 0 is the current microcore	0
seq1	The microcore 1 is the current microcore	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	0	0	1	1	0	0	UcSel	op1	1	0	1

joidr	Jump relative on microcore condition	joidr

Assembler syntax: joidr *Dest UcSel*;

Description:

Configures the jump to relative location on condition.

If the condition defined by the *UcSel* operand is satisfied, the program counter (uPC) is handled such as the next executed instruction is relative destination address.

The jump is relative to the instruction Code RAM location. The destination address is the actual instruction Code RAM location added to the Dest operand value. This 5-bit value is a two's complemented number. The MSB is the sign. So *Dest* operand value is in the range of {-16, 15}.

Operands:

Dest – Operand that defines the 5-bit relative destination address in the range of {-16, 15}.

UcSel – Operand that defines the microcore identifier condition

Operand Label	Operand Description	Operand Binary Value
seq0	The microcore 0 is the current microcore	0
seq1	The microcore 1 is the current microcore	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	1	1	0	1	UcSel	Dest				

joslf Jump far on start condition joslf

Assembler syntax: joslf *op1 StSel*;

Description:

Configures the jump to absolute location on condition.

If the condition defined by the *StSel* operand is satisfied, the program counter (uPC) is handled such as the next executed instruction is located into the destination address contained in one of the jump registers.

The destination address defined by the *op1* register is any of the absolute Code RAM location.

Operands:

op1 – One of the register listed in the operand subset *JpReg*

StSel – Operand that defines the start condition to be satisfied to enable the jump far

Operand Label	Operand Description	Operand Binary Value
none	No start latched	000000
start1	Start 1 latched	000001
start2	Start 2 latched	000010
start12	Start 1,2 latched	000011
start3	Start 3 latched	000100
start13	Start 1,3 latched	000101
start23	Start 2,3 latched	000110
start123	Start 1,2,3 latched	000111
start4	Start 4 latched	001000
start14	Start 1,4 latched	001001
start24	Start 2,4 latched	001010
start124	Start 1,2,4 latched	001011
start34	Start 3,4 latched	001100
start134	Start 1,3,4 latched	001101
start234	Start 2,3,4 latched	001110
start1234	Start 1,2,3,4 latched	001111
start5	Start 5 latched	010000
start15	Start 1,5 latched	010001
start25	Start 2,5 latched	010010
start125	Start 1,2,5 latched	010011
start35	Start 3,5 latched	010100
start135	Start 1,3,5 latched	010101
start235	Start 2,3,5 latched	010110

Operand Label	Operand Description	Operand Binary Value
start1235	Start 1,2,3,5 latched	010111
start45	Start 4,5 latched	011000
start145	Start 1,4,5 latched	011001
start245	Start 2,4,5 latched	011010
start1245	Start 1,2,4,5 latched	011011
start345	Start 3,4,5 latched	011100
start1345	Start 1,3,4,5 latched	011101
start2345	Start 2,3,4,5 latched	011110
start12345	Start 1,2,3,4,5 latched	011111
start6	Start 6 latched	100000
start16	Start 1,6 latched	100001
start26	Start 2,6 latched	100010
start126	Start 1,2,6 latched	100011
start36	Start 3,6 latched	100100
start136	Start 1,3,6 latched	100101
start236	Start 2,3,6 latched	100110
start1236	Start 1,2,3,6 latched	100111
start46	Start 4,6 latched	101000
start146	Start 1,4,6 latched	101001
start246	Start 2,4,6 latched	101010
start1246	Start 1,2,4,6 latched	101011
start346	Start 3,4,6 latched	101100
start1346	Start 1,3,4,6 latched	101101
start2346	Start 2,3,4,6 latched	101110
start12346	Start 1,2,3,4,6 latched	101111
start56	Start 5,6 latched	110000
start156	Start 1,5,6 latched	110001
start256	Start 2,5,6 latched	110010
start1256	Start 1,2,5,6 latched	110011
start356	Start 3,5,6 latched	110100
start1356	Start 1,3,5,6 latched	110101
start2356	Start 2,3,5,6 latched	110110
start12356	Start 1,2,3,5,6 latched	110111
start456	Start 4,5,6 latched	111000
start1456	Start 1,4,5,6 latched	111001
start2456	Start 2,4,5,6 latched	111010
start12456	Start 1,2,4,5,6 latched	111011
start3456	Start 3,4,5,6 latched	111100
start13456	Start 1,3,4,5,6 latched	111101

Operand Label	Operand Description	Operand Binary Value
start23456	Start 2,3,4,5,6 latched	111110
start123456	Start 1,2,3,4,5,6 latched	111111

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	StSel						op1	0	0	0	1

joslr Jump relative on start condition joslr

Assembler syntax: joslr *Dest StSel*;

Description:

Configures the jump to relative location on condition.

If the condition defined by the *StSel* operand is satisfied, the program counter (uPC) is handled such as the next executed instruction is relative destination address.

The jump is relative to the instruction Code RAM location. The destination address is the actual instruction Code RAM location added to the Dest operand value. This 5-bit value is a two's complemented number. The MSB is the sign. So *Dest* operand value is in the range of {-16, 15}.

Operands:

Dest – Operand that defines the 5-bit relative destination address in the range of {-16, 15}.

StSel – Operand that defines the start condition to be satisfied to enable the jump far

Operand Label	Operand Description	Operand Binary Value
none	No start latched	000000
start1	Start 1 latched	000001
start2	Start 2 latched	000010
start12	Start 1,2 latched	000011
start3	Start 3 latched	000100
start13	Start 1,3 latched	000101
start23	Start 2,3 latched	000110
start123	Start 1,2,3 latched	000111
start4	Start 4 latched	001000
start14	Start 1,4 latched	001001
start24	Start 2,4 latched	001010
start124	Start 1,2,4 latched	001011
start34	Start 3,4 latched	001100
start134	Start 1,3,4 latched	001101
start234	Start 2,3,4 latched	001110
start1234	Start 1,2,3,4 latched	001111
start5	Start 5 latched	010000
start15	Start 1,5 latched	010001
start25	Start 2,5 latched	010010
start125	Start 1,2,5 latched	010011
start35	Start 3,5 latched	010100

Operand Label	Operand Description	Operand Binary Value
start135	Start 1,3,5 latched	010101
start235	Start 2,3,5 latched	010110
start1235	Start 1,2,3,5 latched	010111
start45	Start 4,5 latched	011000
start145	Start 1,4,5 latched	011001
start245	Start 2,4,5 latched	011010
start1245	Start 1,2,4,5 latched	011011
start345	Start 3,4,5 latched	011100
start1345	Start 1,3,4,5 latched	011101
start2345	Start 2,3,4,5 latched	011110
start12345	Start 1,2,3,4,5 latched	011111
start6	Start 6 latched	100000
start16	Start 1,6 latched	100001
start26	Start 2,6 latched	100010
start126	Start 1,2,6 latched	100011
start36	Start 3,6 latched	100100
start136	Start 1,3,6 latched	100101
start236	Start 2,3,6 latched	100110
start1236	Start 1,2,3,6 latched	100111
start46	Start 4,6 latched	101000
start146	Start 1,4,6 latched	101001
start246	Start 2,4,6 latched	101010
start1246	Start 1,2,4,6 latched	101011
start346	Start 3,4,6 latched	101100
start1346	Start 1,3,4,6 latched	101101
start2346	Start 2,3,4,6 latched	101110
start12346	Start 1,2,3,4,6 latched	101111
start56	Start 5,6 latched	110000
start156	Start 1,5,6 latched	110001
start256	Start 2,5,6 latched	110010
start1256	Start 1,2,5,6 latched	110011
start356	Start 3,5,6 latched	110100
start1356	Start 1,3,5,6 latched	110101
start2356	Start 2,3,5,6 latched	110110
start12356	Start 1,2,3,5,6 latched	110111
start456	Start 4,5,6 latched	111000
start1456	Start 1,4,5,6 latched	111001
start2456	Start 2,4,5,6 latched	111010
start12456	Start 1,2,4,5,6 latched	111011

Operand Label	Operand Description	Operand Binary Value
start3456	Start 3,4,5,6 latched	111100
start13456	Start 1,3,4,5,6 latched	111101
start23456	Start 2,3,4,5,6 latched	111110
start123456	Start 1,2,3,4,5,6 latched	111111

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	StSel						Dest				

jsrf Jump far on status register bit condition jsrf

Assembler syntax: jsrf *op1 SrSel Pol*;

Description:

Configures the jump to absolute location on status register condition.

If the condition defined by the *SrSel* operand is satisfied according to the polarity *Pol,* the program counter (uPC) is handled such as the next executed instruction is located into the destination address contained in one of the jump registers.

The destination address defined by the *op1* register is any of the absolute Code RAM location.

Operands:

op1 – One of the register listed in the operand subset *JpReg*

SrSel – Operand that defines the status register condition (Ctrl_reg_uc0 and Ctrl_reg_uc1 registers (0x101, 0x102, 0x121, 0x122)) that trigs the jump

Operand Label	Operand Description	Operand Binary Value
b0	Status register bit 0 (LSB)	0000
b1	Status register bit 1	0001
b2	Status register bit 2	0010
b3	Status register bit 3	0011
b4	Status register bit 4	0100
b5	Status register bit 5	0101
b6	Status register bit 6	0110
b7	Status register bit 7	0111
b8	Status register bit 8	1000
b9	Status register bit 9	1001
b10	Status register bit 10	1010
b11	Status register bit 11	1011
b12	Status register bit 12	1100
b13	Status register bit 13	1101
b14	Status register bit 14	1110
b15	Status register bit 15 (MSB)	1111

Pol – Operand that defines the active polarity for the selected bit

Operand Label	Operand Description	Operand Binary Value
low	Active condition if the selected bit is ‘0’	0
high	Active condition if the selected bit is ‘1’	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	1	Pol	SrSel				op1	0	1	0	1

jsrr Jump relative on status register bit condition **jsrr**

Assembler syntax: jsrr *Dest SrSel Pol*;

Description:

Configures the jump to the relative location of the status register condition.

If the condition defined by the *SrSel* operand is satisfied according to the polarity *Pol,* the program counter (uPC) is handled such as the next executed instruction is relative destination address.

The jump is relative to the instruction Code RAM location. The destination address is the actual instruction Code RAM location added to the Dest operand value. This 5-bit value is a two's complemented number. The MSB is the sign. So *Dest* operand value is in the range of {-16, 15}.

Operands:

Dest – Operand that defines the 5-bit relative destination address in the range of {-16, 15}.

SrSel – Operand that defines the status register condition (Ctrl_reg_uc0 and Ctrl_reg_uc1 registers (0x101, 0x102, 0x121, 0x122)) that trigs the jump

Operand Label	Operand Description	Operand Binary Value
b0	Status register bit 0 (LSB)	0000
b1	Status register bit 1	0001
b2	Status register bit 2	0010
b3	Status register bit 3	0011
b4	Status register bit 4	0100
b5	Status register bit 5	0101
b6	Status register bit 6	0110
b7	Status register bit 7	0111
b8	Status register bit 8	1000
b9	Status register bit 9	1001
b10	Status register bit 10	1010
b11	Status register bit 11	1011
b12	Status register bit 12	1100
b13	Status register bit 13	1101
b14	Status register bit 14	1110
b15	Status register bit 15 (MSB)	1111

Pol – Operand that defines the active polarity for the selected bit

Operand Label	Operand Description	Operand Binary Value
low	Active condition if the selected bit is ‘0’	0
high	Active condition if the selected bit is ‘1’	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	0	Pol	SSel				Dest				

jtsf Jump far to subroutine **jtsf**

Assembler syntax: jtsf *op1*;

Description:

Configures the jump on subroutine to absolute location

The program counter (uPC) is handled such as the next executed instruction is located into the destination address contained in one of the jump registers.

When jump to subroutine is called, the current program counter value (uPC) is stored into the auxiliary register (aux) to handle end of subroutine return.

The destination address defined by the *op1* register is any of the absolute Code RAM location.

Operands:

op1 – One of the register listed in the operand subset *JpReg*

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	0	0	1	1	0	1	1	op1	1	0	1

jtsr Jump relative to subroutine jtsr

Assembler syntax: jtsr *Dest ;*

Description:

Configures the jump to subroutine to relative location on condition.

When jump to subroutine is called, the current program counter value (uPC) is stored into the auxiliary register (aux) to handle end of subroutine return.

The jump is relative to the instruction Code RAM location. The destination address is the actual instruction Code RAM location added to the Dest operand value. This 5-bit value is a two's complemented number. The MSB is the sign. So *Dest* operand value is in the range of {-16, 15}.

Operands:

Dest – Operand that defines the 5-bit relative destination address in the range of {-16, 15}.

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	1	1	0	0	1	Dest				

ldca Load counter from ALU register and set outputs ldca

Assembler syntax: ldca *Rst Sh1 Sh2 op1 Eoc*;

Description:

Loads one of the four end of count register (eoc1, eoc2, eoc3, eoc4) defined by the operand *Eoc* with a value stored in a ALU register *op1* and sets the outputs defined by the shortcut *Sh1* and *Sh2*.

Operands:

Rst – Operand (Boolean) that defines if the selected counter value must be reset to zero or must be unchanged.

Operand Label	Operand Description	Operand Binary Value
_rst	The counter value is maintained, only the end of counter is modified	0
rst	The counter value is reset to zero and start to count from zero	1

Sh1, Sh2– Operands that set the first and second shortcuts related to the corresponding outputs. The output shortcuts are defined using the *dfsct* instruction.

Operand Label	Operand Description	Operand Binary Value
keep	No changes, maintains the previous setting	00
off	Automatic diagnosis disabled	01
on	Automatic diagnosis enabled	10
toggle	Reverse the previous setting	11

1 – One of the register listed in the operand subset *AluReg.*

Eoc– Operand that defines the end of count targeted among the four counters available.

Operand Label	Operand Description	Operand Binary Value
c1	Register eoc1	00
c2	Register eoc2	01
c3	Register eoc3	10
c4	Register eoc4	11

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	Rst	Sh1		Sh2		Eoc		op1			1	0	0

ldcd Load counter from Data RAM and set outputs ldcd

Assembler syntax: ldcd *Rst Ofs Sh1 Sh2 Dram Eoc*;

Description:

Loads one of the four end of count register (eoc1, eoc2, eoc3, eoc4) *Eoc* with a value stored in the 6-bit Data RAM address *Dram* and sets the outputs defined by the shortcut *Sh1* and *Sh2*.

The operand *Dram* can be identified with a univocal label. The compiler automatically substitutes the 'define' label (if used) with the suitable Data RAM address.

The Data RAM address is accessed according to the Boolean operand *Ofs* using the:

Immediate addressing mode (IM).

Indexed addressing mode (XM). In that case address base is added the address

Dram. The address base is set using the *stab* instructions.

Operands:

Rst – Operand (Boolean) that defines if the selected counter value must be reset to zero or must be unchanged.

Operand Label	Operand Description	Operand Binary Value
_rst	The counter value is maintained, only the end of counter is modified	0
rst	The counter value is reset to zero and start to count from zero	1

Ofs– Operands that set Data RAM addressing mode

Operand Label	Operand Description	Operand Binary Value
_ofs	Data RAM immediate addressing mode (IM)	0
ofs	Data RAM indexed addressing mode (XM)	1

Sh1, Sh2– Operands that set the first and second shortcuts related to the corresponding outputs. The output shortcuts are defined using the *dfsct* instruction.

Operand Label	Operand Description	Operand Binary Value
keep	No changes, maintains the previous setting	00
off	Automatic diagnosis disabled	01
on	Automatic diagnosis enabled	10
toggle	Reverse the previous setting	11

Dram– Operand that defines the 6-bit DRAM address

Eoc– Operand that defines the end of count targeted among the four counters available.

Operand Label	Operand Description	Operand Binary Value
c1	Register eoc1	00
c2	Register eoc2	01
c3	Register eoc3	10
c4	Register eoc4	11

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	Ofs	Rst	Sh1		Sh2		Eoc		Dram					

ldirh Load 8-MSB ir register ldirh

Assembler syntax: ldirh *Value8 RstH*;

Description:

Loads the *Value8* data in the 8-MSB of the immediate register (ir).

Operands:

Value8 – Operand that defines the 8-bit value to be loading into the 8-MSB of the immediate register

RstH– Operand (Boolean) that defines if set to zero the 8-MSB of the immediate register

Operand Label	Operand Description	Operand Binary Value
_rst	No change on the ir[15:8]	0
rst	Set the Zero the ir[15:8]	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	RstH	Value8								1	1

ldirl Load 8-LSB ir register ldirl

Assembler syntax: ldirl *Value8 RstL*;

Description:

Loads the *Value8* data in the 8-LSB of the immediate register (ir).

Operands:

Value8 – Operand that defines the 8-bit value to be loading into the 8-MSB of the immediate register

RstL– Operand (Boolean) that defines if set to zero the 8-LSB of the immediate register

Operand Label	Operand Description	Operand Binary Value
_rst	No change on the ir[7:0]	0
rst	Set the Zero the ir[7/0]	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	RstL	Value8								1	0

ldjr1 Load jump register 1 **ldjr1**

Assembler syntax: ldjr1 *Value10*;

Description:

Loads the *Value10* data in the 16-bit jump register 1 (jr1).

The operand *Value10* can be replaced by a label. The compiler automatically substitutes the label (if used) with the defined value.

Operands:

Value10 – Operand that defines the 10-bit value to be loading into the jump register 1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Value10										0	0

ldjr2 Load jump register 2 **ldjr2**

Assembler syntax: ldjr2 *Value10*;

Description:

Loads the *Value10* data in the 16-bit jump register 2 (jr2).

The operand *Value10* can be replaced by a label. The compiler automatically substitutes the label (if used) with the defined value.

Operands:

Value10 – Operand that defines the 10-bit value to be loading into the jump register 2

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	Value10										0	0

load Load data from Data RAM to register load

Assembler syntax: load *Dram op1 Ofs*;

Description:

Loads the data from the Data RAM at the address defined by the Dram operand to the *op1* register.

The operand *Dram* can be identified with a univocal label. The compiler automatically substitutes the ‘define’ label (if used) with the suitable Data RAM address.

The Data RAM address is accessed according to the Boolean operand *Ofs* using the:

Immediate addressing mode (IM).

Indexed addressing mode (XM). In that case, address base is added the address *Dram*. The address base is set using the *stab* instructions.

Operands:

Dram– Operand that defines the 6-bit Data RAM address

op1 – One of the register listed in the operand subset *UcReg*

Ofs– Operands that set data RAM addressing mode

Operand Label	Operand Description	Operand Binary Value
_ofs	Data RAM immediate addressing mode (IM)	0
ofs	Data RAM indexed addressing mode (XM)	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	Dram						op1					Ofs	1	0

mul Two ALU registers multiplication to reg32 mul

Operation: (Source1) x (Source2) => (Destination)

Assembler syntax: mul *op1 op2*;

Description:

Multiplies the value contained in *the op1* register with the value contained in *op2* register and places the result in the reg32 register. The reg32 register is the concatenation of the multiplication result registers mh and ml:

mh contains the 16-MSB

ml contains the 16-MSB

The multiplication requires 17 ck clock cycles to be completed.

Operands:

op1 – One of the register listed in the operand subset *AluGprIrReg*

op2 – One of the register listed in the operand subset *AluGprIrReg*

Condition Register:

MO - Multiplication shift overflow

ML - Multiplication shift precision loss

OD –Operation complete

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	0	1	1	0		op2			op1	

muli ALU register multiplication with immediate value to reg32 muli

Operation: (Source) x Immediate value => (Destination)

Assembler syntax: muli *op1 Imm*;

Description:

Multiplies the value contained in *the op1* register with the immediate value *Imm* and places the result in the *r*eg32 register. The reg32 register is the concatenation of the multiplication result registers mh and ml:

mh contains the 16-MSB

ml contains the 16-LSB

The multiplication requires 17 ck clock cycles to be completed.

Operands:

op1 – One of the register listed in the operand subset *AluGprIrReg*

Imm –The *Imm* 4-bit immediate data register

Condition Register:

MO - Multiplication shift overflow

ML - Multiplication shift precision loss

OD –Operation complete

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	1	1	1	0	Imm				op1		

not Invert ALU register bits not

Operation: (Source) \ => (Source)

Assembler syntax: not *op1*;

Description:

Inverts each bit of the *op1* register and places the result in the *op1* register.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Condition Register:

MN – Mask result is 0x0000

MM - Mask result is 0xFFFF

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	1	0	1	1	0	1	1		op1	

or OR mask on ALU register with immediate register to ALU register or

Operation: (Source) (+) Immediate register => (Source)

Assembler syntax: or *op1 ir*;

Description:

Applies the OR-mask stored in the Immediate Register (ir) to the *op1* register and places the result in the *op1* register.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Condition Register:

MN – Mask result is 0x0000

MM - Mask result is 0xFFFF

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	1	0	1	1	0	0	0	op1		

rdspi SPI read request rdspi

Assembler syntax: rdspi;

Description:

Requests an SPI backdoor read.

The address must previously be defined in the SPI address register *spi_add*.

The *rdspi* instruction requires 2 ck cycle to complete operation. The SPI address register must not be changed on the following instruction, otherwise the operation fails and the read data is dummy.

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	0	1	0	0	0	0	0	0

reqi Software interrupt request **reqi**

Assembler syntax: reqi *id*;

Description:

Requests a software interrupt

At the reqi instruction execution, the Code RAM address currently executed is stored in the interrupt return register corresponding to the 10 LSB of the Ucx_irq_status register (0x10F and 0x12F)

By default, the return address of an interrupt is the line where the code was interrupted. In the case of a software interrupt, the return address is the address where the code was interrupted + 1.

A software interrupt must not be interrupted.

Operands:

Id – Operand that defines the 2-bit software interrupt request identifier.

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	1	0	0	0	1	0	Id	

rfs Return from subroutine rfs

Assembler syntax: rfs;

Description:

Ends a subroutine.

To continue the code execution, the program counter (uPC) is loaded with the content of the auxiliary register (aux) that was automatically updated when the subroutine was called with the instructions *jtsf* and *jtsr*.

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	1	0	0	1	1	1	0	0

rstreg Registers reset rstreg

Assembler syntax: rstreg *TgtBit*;

Description:

Resets single or multiple registers defined by the *TgtBit* operand. The instruction reset bits issued from SPI registers including:

- control register Ctrl_reg_ucX (0x101, 0x102, 0x121, 0x122)
- status register Status_reg_ucX registers (0x105, 0x106, 0x125, 0x126)
- automatic diagnosis register Err_ucXchY (0x162 to 0x169)

Operands:

TgtBit– Operands that defines the registers to be reset.

Operand Label	Operand Description	Operand Binary Value
sr	Reset status bits of the status registers	000
cr	Reset control register	001
sr_diag_halt	Reset status bits, automatic diagnosis register and re-enables the possibility to generate automatic diagnosis interrupts	010
all	Reset status bits, control register, automatic diagnosis register and re-enables the possibility to generate automatic diagnosis interrupts	011
diag_halt	Reset automatic diagnosis register and re-enables the possibility to generate automatic diagnosis interrupts	100
sr_cr	Reset status bits and control register	101
sr_halt	Reset status bits and re-enables the possibility to generate automatic diagnosis interrupts	110
halt	Re-enables the possibility to generate automatic diagnosis interrupts	111

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	0	1	0	0	1	TgtBit		

rstsl Start-latch registers reset rstsl

Assembler syntax: rstsl;

Description:

Resets the Start_latch_ucx register.

This instruction is active only if the Smart Latch Mode is enabled. The smart mode register can be activated by setting the bits smart_start_uc0 and smart_start_uc1 of the Start_config_reg registers (0x104, 0x124).

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	1	0	0	1	1	1	0	1

sh32l Shift left multiplication result register sh32l

Operation: (Source) << factor => (Source)

Assembler syntax: sh32l *op1*;

Description:

Shifts the reg32 register left. The shift is single or multiple according to the *op1* register value (factor).

The reg32 register is the concatenation of the multiplication result registers mh and ml:

- mh contains the 16-MSB
- ml contains the 16-LSB

To be completed, the shift operation requires a number of ck clock cycles corresponding to the *op1* register value.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Condition Register:

SB – Shift out bit

MO - Multiplication shift overflow

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	0	0	1	0	1	op1			1	0	1

sh32li Shift left multiplication result register of immediate value sh32li

Operation: (Source) << Immediate value => (Source)

Assembler syntax: sh32li *Imm*;

Description:

Shifts the reg32 register left. The shift is single or multiple according to the immediate value (factor).

The reg32 register is the concatenation of the multiplication result registers mh and ml:

- mh contains the 16-MSB
- ml contains the 16-LSB

To be completed, the shift operation requires a number of ck clock cycles corresponding to the immediate value.

Operands:

Imm –The *Imm* 4-bit immediate data register

Condition Register:

SB – Shift out bit

MO - Multiplication shift overflow

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	0	0	0	1	Imm				1	0	1

sh32r Shift right multiplication result register sh32r

Operation: (Source) >> factor => (Source)

Assembler syntax: sh32r *op1*;

Description:

Shifts the reg32 register right. The right shift is single or multiple according to the *op1* register value (factor).

The reg32 register is the concatenation of the multiplication result registers mh and ml:

- mh contains the 16-MSB
- ml contains the 16-LSB

To be completed, the shift operation requires a number of ck clock cycles corresponding to the *op1* register value.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Condition Register:

SB – Shift out bit

ML - Multiplication shift precision loss

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	0	0	1	0	0	op1			1	0	1

sh32ri Shift right multiplication result register of immediate value sh32ri

Operation: (Source) >> Immediate value => (Source)

Assembler syntax: sh32ri *Imm*;

Description:

Shifts the reg32 register right. The right shift is single or multiple according to the immediate value.

The reg32 register is the concatenation of the multiplication result registers mh and ml:

- mh contains the 16-MSB
- ml contains the 16-LSB

To be completed, the shift operation requires a number of ck clock cycles corresponding to the immediate value.

Operands:

Imm –The *Imm* 4-bit immediate data register

Condition Register:

SB – Shift out bit

ML - Multiplication shift precision loss

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	0	0	0	0	Imm				1	0	1

shl Shift left ALU register shl

Operation: (Source) << factor => (Source)

Assembler syntax: shl *op1 op2*;

Description:

Shifts the *op1* register left. The shift is single or multiple according to the *op2* register value (factor).

To be completed, the shift operation requires a number of ck clock cycles corresponding to the *op2* register value.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

*op*2– One of the register listed in the operand subset *AluReg*

Condition Register:

SB – Shift out bit

MO - Multiplication shift overflow

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	1	0	1	0	op2			op1		

shl8 Shift left ALU register of 8 bits shl8

Operation: (Source) << 8 => (Source)

Assembler syntax: shl8 *op1*;

Description:

Shifts the *op1* register of 8 positions left.

To be completed, the shift operation requires one ck clock cycles.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Condition Register:

SB – Shift out bit

MO - Multiplication shift overflow

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	1	0	1	1	1	1	1	op1		

shli Shift left the ALU register of immediate value shli

Operation: (Source) << immediate value => (Source)

Assembler syntax: shl *op1 Imm*;

Description:

Shift the *op1* register left. The shift is single or multiple according to the immediate value *Imm.*

To be completed, the shift operation requires a number of ck clock cycles corresponding to the immediate value *Imm*.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Imm –The *Imm* 4-bit immediate data register

Condition Register:

SB – Shift out bit

MO - Multiplication shift overflow

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	0	1	0	Imm				op1		

shls Shift left signed ALU register shls

Operation: (Source) << factor => (Source)

Assembler syntax: shls *op1 op2*;

Description:

Shift the *op1* register left. The shift is single or multiple according to the *op2* register value (factor).

The *op1* register is handled as a two’s complement number. Its MBS (sign bit) is unchanged during the shift operation.

To be completed, the shift operation requires a number of ck clock cycles corresponding to the *op2* register value.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

op1 – One of the register listed in the operand subset *AluReg*

Condition Register:

SB – Shift out bit

MO - Multiplication shift overflow

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	1	0	0	0	op2			op1		

shlsi Shift left signed ALU register of immediate value shlsi

Operation: (Source) << immediate value => (Source)

Assembler syntax: shls *op1 Imm*;

Description:

Shifts the *op1* register left. The shift is single or multiple according to the immediate value *Imm*.

The *op1* register is handled as a two’s complement number. Its MBS (sign bit) is unchanged during the shift operation.

To be completed, the shift operation requires a number of ck clock cycles corresponding to the immediate value *Imm*.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Imm –The *Imm* 4-bit immediate data register

Condition Register:

SB – Shift out bit

MO - Multiplication shift overflow

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	0	0	0	Imm				op1		

shr Shift right ALU register shr

Operation: (Source) >> factor => (Source)

Assembler syntax: shr *op1 op2*;

Description:

Shift the *op1* register right. The shift is single or multiple according to the *op2* register value (factor).

To be completed, the shift operation requires a number of ck clock cycles corresponding to the *op2* register value.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

*op*2– One of the register listed in the operand subset *AluReg*

Condition Register:

SB – Shift out bit

ML - Multiplication shift precision loss

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	1	0	0	1	op2			op1		

shr8 Shift right ALU register of 8 bits shr8

Operation: (Source) >> 8 => (Source)

Assembler syntax: shr8 *op1*;

Description:

Shift the *op1* register of 8 positions right.

To be completed, the shift operation requires one ck clock cycle.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Condition Register:

SB – Shift out bit

ML - Multiplication shift precision loss

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	1	0	1	1	1	1	0	op1		

shri Shift right the ALU register of immediate value shri

Operation: (Source) >> immediate value => (Source)

Assembler syntax: shr *op1 Imm*;

Description:

Shifts the *op1* register right. The shift is single or multiple according to the immediate value *Imm.*

To be completed, the shift operation requires a number of ck clock cycles corresponding to the immediate value *Imm*.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Imm –The *Imm* 4-bit immediate data register

Condition Register:

SB – Shift out bit

ML - Multiplication shift precision loss

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	0	0	1	Imm				op1		

shrs Shift right signed ALU register shrs

Operation: (Source) >> factor => (Source)

Assembler syntax: shrs *op1 op2*;

Description:

Shift the *op1* register right. The shift is single or multiple according to the *op2* register value (factor).

The *op1* register is handled as a two's complement number. Its MBS (sign bit) is unchanged during the shift operation.

To be completed, the shift operation requires a number of ck clock cycles corresponding to the *op2* register value.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

op2 – One of the register listed in the operand subset *AluReg*

Condition Register:

SB – Shift out bit

ML - Multiplication shift precision loss

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	0	1	1	1	op2			op1		

shrsi Shift right signed ALU register of immediate value shrsi

Operation: (Source) >> immediate value => (Source)

Assembler syntax: shrsi *op1 Imm*;

Description:

Shifts the *op1* register right. The shift is single or multiple according to the immediate value *Imm*.

The *op1* register is handled as a two's complement number. Its MBS (sign bit) is unchanged during the shift operation.

To be completed, the shift operation requires a number of ck clock cycles corresponding to the immediate value *Imm*.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Imm –The *Imm* 4-bit immediate data register

Condition Register:

SB – Shift out bit

MO - Multiplication shift overflow

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	1	1	1	1	Imm				op1		

slab Select Data RAM address base slab

Assembler syntax: slab *SelBase*;

Description:

Selects the register that contains the address base used in the data RAM Indexed Addressing Mode (XM).

The reset value of *SelBase* is *reg.*

Operands:

SelBase – Operand that defines the register to be used to determine the data RAM address base

Operand Label	Operand Description	Operand Binary Value
reg	Use the dedicated address base *add_base* register. In this case the address base is defined with the *stab* instruction.	0
ir	Use the ALU ir register as address base	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	1	0	0	0	0	0	1	0	1	SelBase

slfbk Select HS2/4 feedback reference slfbk

Assembler syntax: slfbk *Ref Diag*;

Description:

Selects the feedback reference for both V_{DS} of the high-side pre-drivers 2 and 4.

In addition, this instruction enables the automatic diagnosis.

This operation is successful only if the microcore has the right to drive the related outputs. The drive right is granted by setting the related bits in the Out_acc_ucX_chY (0x184, 0x185, 0x186, 0x187) configuration registers.

The reset of *Ref* value is *boost*.

Operands:

Ref – Operand that defines the feedback reference for both VDS of the high-side pre-drivers 2 and 4.

Operand Label	Operand Description	Operand Binary Value
boost	The both VDS of the high-side pre-drivers 2 and 4 are referred to boost voltage (VBOOST pin)	0
bat	The both VDS of the high-side pre-drivers 2 and 4 are referred to bat voltage (VBATT pin)	1

Diag – Operand that defines the diagnosis status for both V_{DS} of the high-side pre-drivers 2 and 4.

Operand Label	Operand Description	Operand Binary Value
keep	No changes, maintains the previous setting	00
NA	Not Applicable	01
off	Automatic diagnosis disabled	10
on	Automatic diagnosis enabled	11

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	1	0	0	1	0	0	0	Ref	Diag	

slsa Select SPI address slsa

Assembler syntax: slsa *SelSpi*;

Description:

Selects the register that contains the address used on SPI read and write instructions (*drspi* and *wrspi*)

The reset values of *SelSpi* is *reg.*

Operands:

SelSpi – Operand that defines the register containing the SPI address

Operand Label	Operand Description	Operand Binary Value
reg	Use the dedicated address register *spi_add*.	0
ir	Use the ALU ir register as SPI address	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	1	0	0	0	0	0	1	0	0	SelSpi

stab Set Data RAM address base stab

Assembler syntax: stab *AddBase*;

Description:

Loads the address value in the address base register *add_base*.

The address base register is a 6-bit register that contains the address base used in the Data RAM Indexed Addressing Mode (XM).

The operand *AddBase* can be identified with a univocal label. The compiler automatically substitutes the ‘define’ label (if used) with the suitable address.

Operands:

AddBase – Operand that defines the 6-bit register containing the Address Base.

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	0	0	0	AddBase					

stadc Set ADC mode stadc

Assembler syntax: stadc *AdcMode DacTarget*;

Description:

Enables or disables the ADC conversion mode on the specified current measurement block.

The operation is successful only if the microcore has the right to access the related current measurement block. The access right is granted by setting the related bits in the Cur_block_access_1 register (0x188) and Cur_block_access_2 Register (0x189).

The reset value of *AdcMode* is off.

Operands:

AdcMode – Operand that activate the ADC mode on the selected current measurement block

Operand Label	Operand Description	Operand Binary Value
off	The current measurement block compares the current flowing in the actuator with a threshold (nominal behavior).	0
on	The current measurement block performs an analog to digital conversion of the current flowing in the actuator	1

DacTarget – Operand that defines the current measurement block DAC to be set in ADC mode

Operand Label	Operand Description	Operand Binary Value
sssc	DAC of the same microcore same channel	00
ossc	DAC of the other microcore same channel	01
ssoc	DAC of the same microcore other channel	10
osoc	DAC of the other microcore other channel	11

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	0	1	1	0	1	Adc Mode	DacTarget	

stal Set arithmetic logic mode stal

Assembler syntax: stal *ModeAL*;

Description:

Sets the arithmetic logic mode. This mode is the set according to the bits A1 and A0 of the ALU condition register (arith_reg).

ALU operations behavior is affected by the arithmetic logic mode *ModeAL* as described below:

The ALU instruction operands are handled as C-complement number (signed number). If the resulting value exceeds the result register capacity, leads to overflow detection but no saturation.

The ALU instruction operands are handled as C-complement number (signed number). If the resulting value exceeds the result register capacity, it leads to overflow detection and saturation (arith_logic_c2_sat).

The ALU instruction operands are handled as positive number (unsigned number). If the resulting value exceeds the result register capacity it leads to overflow detection but no saturation.

The ALU instruction operands are handled as positive number (unsigned number). If the resulting value exceeds the result register capacity it leads to overflow detection and saturation.

The *ModeAL* reset value is *al3*.

Operands:

ModeAL – Operand that defines the ALU behavior selected

Operand Label	Operand Description	Operand Binary Value
al1	two’s complement number without overflow saturation	00
al2	two’s complement number with overflow saturation	01
al3	Positive number without overflow saturation	10
al4	Positive number with overflow saturation	11

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	0	1	1	1	1	1	ModeAL	

stcrb Set control register bit stcrb

Assembler syntax: stcrb *Logic CrbSel*;

Description:

Sets the logic level value individually with the *Logic* operand of each selected bit *CrbSel* of the control register.

Operands:

Logic – Operand that defines the logic level value

Operand Label	Operand Description	Operand Binary Value
low	Low level	0
high	High level	1

CrbSel – Operand that defines the control register bit to be selected

Operand Label	Operand Description	Operand Binary Value
b8	Control register bit 8	000
b9	Control register bit 9	001
b10	Control register bit 10	010
b11	Control register bit 11	011
b12	Control register bit 12	100
b13	Control register bit 13	101
b14	Control register bit 14	110
b15	Control register bit 15 (MSB)	111

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	1	1	0	1	0	Logic	0	CrbSel		

stcrt Set channel communication register stcrt

Assembler syntax: stcrt *UcId*;

Description:

Each microcore:

shares the ch_rxtx register with the other microcores

can read the shared register of another microcore.

This instruction selects the microcore's shared register that is accessed by the microcore executing the stcrt instruction.

The *UcId* reset value is sssc.

Operands:

UcId – Operand that defines the microcore shared register to be access.

Operand Label	Operand Description	Operand Binary Value
sssc	The microcore that is executing the code	00
ossc	The other microcore in the same channel	01
ssoc	The same microcore in the other channel	10
osoc	The other microcore in the other channel	11

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	1	0	0	0	1	1	UcId	

stdcctl Set DC-DC control mode **stdcctl**

Assembler syntax: stdcctl *ModeDC*;

Description:

Selects if the DCDC must be controlled by the microcore (sync) or perform the automatic current regulation (async) by managing controlling the low-side pre-driver 7.

If automatic mode is selected, the current is regulated between threshold 4l and 4h

The *ModeDC* reset value is sync.

Operands:

ModeDC – Operand that defines the DC-DC control mode

Operand Label	Operand Description	Operand Binary Value
sync	DCDC is controlled by the microcore	0
async	DCDC perform an automatic current control between threshold 4l and 4h	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	0	1	0	0	0	0	1	ModeDC

stdm Set DAC register mode access stdm

Assembler syntax: stdm *ModeDAC*;

Description:

The DAC registers address (DAC Register x in DAC Mode and DAC_4h4neg Register) in the internal data memory map are split in two slices:

- dac_value_x and dac_value_x for the DAC Register x in DAC Mode
- dac_value_4neg, dac_value_4h for the DAC_4h4neg Register.

This instruction selects which slice(s) is accessed.

The dac4h4n_boost address in the internal data memory map can refer to three registers (dac4h value, dac4neg value, dac boost value); this same instruction selects which of the three register is accessed, according to the *ModeDAC* operand.

- dac_boost_access_mode: nothing (for the dac address) or the value of the dac boost (for the dac4h4n_boost address) is accessed
- dac_access_mode/dac4h_access_mode: the dac value (for the dac address) or the dac4h value (for the dac4h4n_boost address) is accessed. the result is available in the 8 lower bits
- offset_access_mode/dac4neg_access_mode: the offset register (for the dac address) or the dac4neg value (for the dac4h4n_boost address) is accessed. the result is available in the 13-8 bits if reading an offset, in the 11-8 bits if reading dac4neg
- full_access_mode/dac4h4n_access_mode: both the dac value and the offset register (for the dac address) or both the dac4h and the dac4n value (for the dac4h4n_boost address) is accessed

The *ModeDAC* reset value is dac.

Operands:

ModeDAC – Operand that defines the DAC access mode

Operand Label	Operand Description	Operand Binary Value
null	dac_bst_access_mde	00
dac	dac_access_mde/dac4h_access_mde	01
offset	offset_access_mde/dac4n_access_mode	10
full	full_access_mode/ dac4h4n_access_mode	11

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	1	0	0	0	0	1	ModeDC	

stdrm Set Data RAM read mode stdrm

Assembler syntax: stdrm *ModeDRM*;

Description:

Sets the Data RAM read mode.

The possible read modes according to the *ModeDRM* operand are:

- dram_word_mode: all 16 bits are accessed
- dram_lowbyte _mode: only the 8 LSBs of the source Data RAM are accessed. The result is available in the 8 lower bits of the destination register. The upper 8 bits of the destination register is set to 0x00.
- dram_highbyte_mode: only the 8 MSBs of the source Data RAM are accessed. The result is available in the 8 lower bits of the destination register. The upper 8 bits of the destination register is set to 0x00.
- dram_swapbyte_mode: the 8 LSBs and 8 MSBs of the source dram are accessed swapped and is available at the destination register.

This read mode is valid after the *load* and *ldcd* instructions following this *stdrm* instruction.

The *ModeDRM* reset value is word.

Operands:

ModeDRM – Operand that defines the Data RAM read access

Operand Label	Operand Description	Operand Binary Value
word	dram_word_mode	00
low	dram_lowbyte _mode:	01
high	dram_highbyte_mode	10
swap	dram_swapbyte_mode	11

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	1	0	0	0	0	0	ModeDRM	

steoa Set end of actuation mode steoa

Assembler syntax: steoa *Mask Switch*;

Description:

Enables or disables the end of actuation mode for all the high-side pre-drivers that the microcore is enabled to drive by means of the *Switch* operand.

The V_{SRC} threshold monitoring of the related pre-drivers can be disabled by setting the operand *Mask*

The Mask default value is *nomask.*

The *Switch* default value is *bsoff.*

Operands:

Mask – Operand that set the V_{DS} threshold mask

Operand Label	Operand Description	Operand Binary Value
nomask	V_{SRC} threshold monitoring of the selected HS is unchanged	0
mask	V_{SRC} threshold monitoring of the selected HS is masked to zero	1

Switch – Operand that set the end of actuation mode

Operand Label	Operand Description	Operand Binary Value
keep	Maintain the previous values	00
bsoff	Bootstrap switch is forced off	11
bson	Bootstrap switch can be enabled even if no low-side pre-driver is switched on	01
bsneutral	Bootstrap control is not affected	10

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	0	0	1	0	0	Mask	Switch	

stf Set flag stf

Assembler syntax: stf *Logic FlgSel*;

Description:

Sets the logic level value with the Boolean *Logic* of the selected flag. The flag is selected according the *FlgSel* operand.

Operands:

Logic – Operand that defines the logic level value

Operand Label	Operand Description	Operand Binary Value
low	Low level	0
high	High level	1

FlgSel – Operand that defines the flag bit to be selected

Operand Label	Operand Description	Operand Binary Value
b0	Flag bit 0	0000
b1	Flag bit 1	0001
b2	Flag bit 2	0010
b3	Flag bit 3	0011
b4	Flag bit 4	0100
b5	Flag bit 5	0101
b6	Flag bit 6	0110
b7	Flag bit 7	0111
b8	Flag bit 8	1000
b9	Flag bit 9	1001
b10	Flag bit 10	1010
b11	Flag bit 11	1011
b12	Flag bit 12	1100
b13	Flag bit 13	1101
b14	Flag bit 14	1110
b15	Flag bit 15	1111

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	0	1	1	1	Logic	FlgSel			

stfw Set freewheeling mode **stfw**

Assembler syntax: stfw *FwMode*;

Description:

Defines the freewheeling output modes. Freewheeling control is automatic or manual according to the *FwMode* operand.

The *FwMode* operand is a Boolean that defines the control mode:

- if Shortcut1 is HS1, then LS5 is set as freewheeling pre-driver
- if Shortcut1 is HS2, then LS6 is set as freewheeling pre-driver
- if Shortcut1 is HS3, then LS7 is set as freewheeling pre-driver
- if Shortcut1 is HS4, then HS5 is set as freewheeling pre-driver
- if Shortcut1 is HS5, then LS4 is set as freewheeling pre-driver.

The shortcuts are set using the *dfsct* instruction.

This operation is successful only if the microcore has the right to drive the output related to freewheeling. The drive right is granted by setting the related bits in the Out_acc_ucX_chY (0x184, 0x185, 0x186, 0x187) configuration registers.

The *FwMode* reset value is manual.

Operands:

FwMode – Operand that defines the freewheeling mode

Operand Label	Operand Description	Operand Binary Value
manual	Freewheeling manual control	0
auto	Freewheeling automatic control	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	1	0	0	1	0	0	0	FwMode

stgn Set current measure operational amplifier gain stgn

Assembler syntax: stgn *Gain OpAmp*;

Description:

Sets the gain of an operational amplifier with the *Gain* operand used to measure the current flowing through the actuator sense resistor. The operational amplifier is selected according to the *OpAmp* operand.

The operation is successful only if the microcore has the right to access the related current measurement block. The access right is granted by setting the related bits in the Cur_block_access_1 register (0x188) and Cur_block_access_2 Register (0x189).

The *Gain* reset value is gain 5.8.

Operands:

Gain – Operand that defines the current measure operational amplifier gain

Operand Label	Operand Description	Operand Binary Value
gain5.8	Operational amplifier gain set to 5.8	00
gain8.7	Operational amplifier gain set to 8.7	01
gain12.6	Operational amplifier gain set to 12.6	10
gain19.3	Operational amplifier gain set to 19.3	11

OpAmp – Operand that defines the current measure operational amplifier gain to be set

Operand Label	Operand Description	Operand Binary Value
sssc	Current measure operational amplifier of the same microcore same channel	00
ossc	Current measure operational amplifier of the other microcore same channel	01
ssoc	Current measure operational amplifier of the same microcore other channel	10
osoc	Current measure operational amplifier of the other microcore other channel	11

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	0	0	1	1	Gain		OpAmp	

stirq Set IRQB pin **stirq**

Assembler syntax: stirq *Logic;*

Description:

Set the IRQB output pin

The *Logic* reset value is high.

Operands:

Logic – Operand that defines the logic level of the IRQB pin

Operand Label	Operand Description	Operand Binary Value
low	Low level	0
high	High level	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	1	0	0	1	0	0	1	Logic

sto Set single pre-driver output sto

Assembler syntax: sto *OutSel Out*;

Description:

Sets the state with the *Out* operand for the selected output according to the *OutSel* operand.

The operation is successful only if the microcore has the right to drive the related outputs. The drive right is granted by setting the related bits in the Out_acc_ucX_chY (0x184, 0x185, 0x186, 0x187) configuration registers.

Operands:

OutSel – Operand that defines the handled output

Operand Label	Operand Description	Operand Binary Value
hs1	High-side pre-driver 1	0000
hs2	High-side pre-driver 2	0001
hs3	High-side pre-driver 3	0010
hs4	High-side pre-driver 4	0011
hs5	High-side pre-driver 5	0100
ls1	Low-side pre-driver 1	0101
ls2	Low-side pre-driver 2	0110
ls3	Low-side pre-driver 3	0111
ls4	Low-side pre-driver 4	1000
ls5	Low-side pre-driver 5	1001
ls6	Low-side pre-driver 6	1010
ls7	Low-side pre-driver 7	1011
undef	Undefined	1100

Out – Operand that set output state

Operand Label	Operand Description	Operand Binary Value
keep	No changes, maintains the previous setting	00
off	Output disabled	01
on	Output enabled	10
toggle	Reverse the previous setting	11

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	0	1	0	OutSel				Out	

stoc Set offset compensation stoc

Assembler syntax: stoc *Ctrl DacTraget*;

Description:

Enables or disables the offset compensation with the operand *Ctrl* on the current measurement block specified according to the *DacTarget* operand.

The operation is successful only if the microcore has the right to access the related current measurement block. The access right is granted by setting the related bits in the Cur_block_access_1 register (0x188) and Cur_block_access_2 Register (0x189).

The *Ctrl* reset value is *off* for all current measurement blocks.

Operands:

Ctrl – Operands that set offset compensation state

Operand Label	Operand Description	Operand Binary Value
off	Disable the offset compensation	0
on	Enable the offset compensation	1

DacTarget – Operand that defines the current measurement block

Operand Label	Operand Description	Operand Binary Value
sssc	DAC of the same microcore same channel	00
ossc	DAC of the other microcore same channel	01
ssoc	DAC of the same microcore other channel	10
osoc	DAC of the other microcore other channel	11

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	0	1	0	1	0	Ctrl	DacTarget	

store Store register data in Data RAM store

Assembler syntax: store *op1 Dram Ofs*;

Description:

Copies the content of the *op1* source register in a Data RAM line defined by the 6-bit Data RAM address *Dram.*

The operand *Dram* can be identified with a univocal label. The compiler automatically substitutes the ‘define’ label (if used) with the suitable Data RAM address.

The Data RAM address is accessed according to the Boolean operand *Ofs* using the:

- Immediate addressing mode (IM).
- Indexed addressing mode (XM). In that case,
- the address base is added to the address *Dram*. The address base is set using the *stab* instructions.

Operands:

op1 – One of the register listed in the operand subset *UcReg*

Dram– Operand that defines the 6-bit DRAM address

Ofs– Operands that set data RAM addressing mode

Operand Label	Operand Description	Operand Binary Value
_ofs	Data RAM immediate addressing mode (IM)	0
ofs	Data RAM indexed addressing mode (XM)	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	Ofs	op1					Dram						0	1

stos Set pre-driver output shortcuts stos

Assembler syntax: stos *Out1 Out2 Out3*;

Description:

Sets the state of three outputs *Out1, Out2* and *Out3* previously defined as shortcuts with the *dfsct* instruction.

The operation is successful only if the microcore has the right to drive the related outputs. The drive right is granted by setting the related bits in the Out_acc_ucX_chY (0x184, 0x185, 0x186, 0x187) configuration registers.

Operands:

Out1, Out2, and Out3 – Operands that set output state

Operand Label	Operand Description	Operand Binary Value
keep	No changes, maintains the previous setting	00
off	Output disabled	01
on	Output enabled	10
toggle	Reverse the previous setting	11

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	0	0	1	Out1		Out12		Out13	

stslew Set pre-driver output slew rate mode **stslew**

Assembler syntax: stslew *SlMode*;

Description:

Defines the outputs slew rate mode with the Boolean *SlMode*.

The operation is successful only if the microcore has the right to drive the related outputs. The drive right is granted by setting the related bits in the Out_acc_ucX_chY (0x184, 0x185, 0x186, 0x187) configuration registers.

The *SlMode* reset value is normal.

When switching the slew-rate from slow to fast, the new slew-rate is valid after typically 1ck cycle (166 ns considering f_{ck} = 6.0 MHz).

When switching from fast to slow, it takes typically four ck cycles (666 ns considering f_{ck} = 6.0 MHz) until the new slew-rate is effective.

Operands:

SlMode – Operands that set outputs slew rate mode

Operand Label	Operand Description	Operand Binary Value
normal	The outputs slew rate is set by an SPI register	0
fast	The outputs slew rate is the highest one	1

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	1	0	0	1	0	1	1	SlMode

stsrb Set status register bit stsrb

Assembler syntax: stsrb *Logic SrbSel*;

Description:

Sets individually the logic level value with the *Logic* operand of each selected bit *SrbSel* of the status register.

Operands:

Logic – Operand that defines the logic level value

Operand Label	Operand Description	Operand Binary Value
low	Low level	0
high	High level	1

SrbSel – Operand that defines the status register bit to be selected

Operand Label	Operand Description	Operand Binary Value
b0	Status register bit 0 (LSB)	0000
b1	Status register bit 1	0001
b2	Status register bit 2	0010
b3	Status register bit 3	0011
b4	Status register bit 4	0100
b5	Status register bit 5	0101
b6	Status register bit 6	0110
b7	Status register bit 7	0111
b8	Status register bit 8	1000
b9	Status register bit 9	1001
b10	Status register bit 10	1010
b11	Status register bit 11	1011
b12	Status register bit 12	1100
b13	Status register bit 13	1101
b14	Status register bit 14	1110
b15	Status register bit 15 (MSB)	1111

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	1	1	0	0	0	Logic	SrbSel			

sub Two ALU registers subtraction to ALU register sub

Operation: (Source1) - (Source2) => (Destination)

Assembler syntax: sub *op1 op2 res*;

Description:

Subtracts the value contained in *the op1* register to the value contained in *op2* register and places the result in the *res* register.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

op2 – One of the register listed in the operand subset *AluReg*

res – One of the register listed in the operand subset *AluReg*

Condition Register:

RZ - Addition or subtraction result is zero

RS - Addition or subtraction result is negative

UU - Unsigned underflow

UO - Unsigned overflow

SU - Signed underflow

SO - Signed overflow

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	0	res			1	op2			op1		

subi ALU register subtraction with immediate value to ALU register subi

Operation: (Source) - Immediate value => (Destination)

Assembler syntax: subi *op1 Imm res*;

Description:

Subtracts the value contained in the *Imm* register to the value contained in *the op1* register and places the result in the *res* register.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Imm –The *Imm* 4-bit immediate data register

res – One of the register listed in the operand subset *AluReg*

Condition Register:

RZ - Addition or subtraction result is zero

RS - Addition or subtraction result is negative

UU - Unsigned underflow

UO - Unsigned overflow

SU - Signed underflow

SO - Signed overflow

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1		res				Imm			op1	

swap Swap bytes inside ALU register **swap**

Operation: (Source)[0:7] < => (Source)[8:15]

Assembler syntax: swap *op1*;

Description:

Swaps the high byte and the low byte of the register *op1*.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	0	1	1	0	0	op1		

toc2 Integer to two’s complement conversion in ALU register toc2

Assembler syntax: toc2 *op1*;

Description:

Converts the integer value contained in *op1* register to two’s complement format.

If the conversion bit CS in the arithmetic condition register arith_reg is zero, the *toc2* instruction set the operand register MSB to zero.

If the conversion bit is one, then it returns the 2's complement of the operand (bits[14:0] only) register *op1* and set the MSB to one.
Operands:

op1 – One of the register listed in the operand subset *AluReg*

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	1	1	0	0	1	1	0	op1		

toint two’s complement to integer conversion in ALU register toint

Assembler syntax: toint *op1 Rst*;

Description:

Convert the two’s complement value contained in *op1* register to integer format.

The *toint* instruction retains the original value in the operand register *op1* when its MSB bit is zero.

If the MSB is 1, then it returns the 2's complement of the operand register (op1[14:0]).

The *toint* instruction also saves the MSB of the operand *op1* in the conversion bit CS of the arithmetic condition register arith_reg.

The MSB of the operand is either XORed with the existing conversion bit CS of the ALU condition register (if the instruction is called with the **_***rst* parameter) or replaces it (if the instruction is called with the *rst* parameter).

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Rst – Operand that defines if the conversion bit CS of the ALU condition register is reset

Operand Label	Operand Description	Operand Binary Value
_rst	The existing conversion bit CS is XORed with the *op1* MSB	0
rst	The existing conversion bit CS is set according to the *op1* MSB	1

Condition Register:

CS - Last conversion sign

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	1	1	0	0	1	0	Rst	op1		

wait Wait until condition satisfied **wait**

Assembler syntax: wait *WaitMask ;*

Description:

Stops the program counter (uPC) incrementing and waits until at least one of the enabled wait conditions is satisfied. When one of the conditions is satisfied, the program counter is moved to the corresponding destination.

The possible wait conditions, along with the corresponding destinations, are stored in the wait table by means of the *cwer* and *cwef* instructions.

The active wait table rows are enabled according to the *WaitMask* 5-bit operand.

Operands:

WaitMask – Operand that defines the active wait table rows

Operand Label	Operand Description	Operand Binary Value
always	No wait table row enabled. Infinite loop	00000
row1	Wait table row 1 enabled	00001
row2	Wait table row 2 enabled	00010
row12	Wait table row 1,2 enabled	00011
row3	Wait table row 3 enabled	00100
row13	Wait table row 1,3 enabled	00101
row23	Wait table row 2,3 enabled	00110
row123	Wait table row 1,2,3 enabled	00111
row4	Wait table row 4 enabled	01000
row14	Wait table row 1,4 enabled	01001
row24	Wait table row 2,4 enabled	01010
row124	Wait table row 1,2,4 enabled	01011
row34	Wait table row 3,4 enabled	01100
row134	Wait table row 1,3,4 enabled	01101
row234	Wait table row 2,3,4 enabled	01110
row1234	Wait table row 1,2,3,4 enabled	01111
row5	Wait table row 5 enabled	10000
row15	Wait table row 1,5 enabled	10001
row25	Wait table row 2,5 enabled	10010
row125	Wait table row 1,2,5 enabled	10011
row35	Wait table row 3,5 enabled	10100
row135	Wait table row 1,3,5 enabled	10101
row235	Wait table row 2,3,5 enabled	10110
row1235	Wait table row 1,2,3,5 enabled	10111

Operand Label	Operand Description	Operand Binary Value
row45	Wait table row 4,5 enabled	11000
row145	Wait table row 1,4,5 enabled	11001
row245	Wait table row 2,4,5 enabled	11010
row1245	Wait table row 1,2,4,5 enabled	11011
row345	Wait table row 3,4,5 enabled	11100
row1345	Wait table row 1,3,4,5 enabled	11101
row2345	Wait table row 2,3,4,5 enabled	11110
row12345	Wait table row 1,2,3,4,5 enabled	11111

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	0	0	0	WaitMask				

wrspi SPI write request **wrspi**

Assembler syntax: wrspi;

Description:

Requests an SPI backdoor write.

The address must previously be defined in the SPI address register *spi_add*.

The data must previously be defined in the SPI address register *spi_data* register.

The *wrspi* instruction requires 2 ck cycles to complete operation. The SPI address register and SPI data register must not be changed on the following instruction, otherwise the operation fails and the written data is dummy.

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	1	0	1	0	0	0	0	0	1

xor XOR-mask on ALU register with the immediate register to ALU register xor

Operation: (Source) **+** Immediate register => (Source)

Assembler syntax: xor *op1*;

Description:

Applies the XOR-mask contained into the *Ir* register to the value contained in the *op1* register and places the result in the *op1* register. The initial data stored in the *op1* register is loss.

Operands:

op1 – One of the register listed in the operand subset *AluReg*

Ir –The ALU immediate register

Condition Register:

MM - Mask result is 0x0000

MN - Mask result is 0xFFFF

Instruction format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	1	0	1	1	0	1	0	op1		

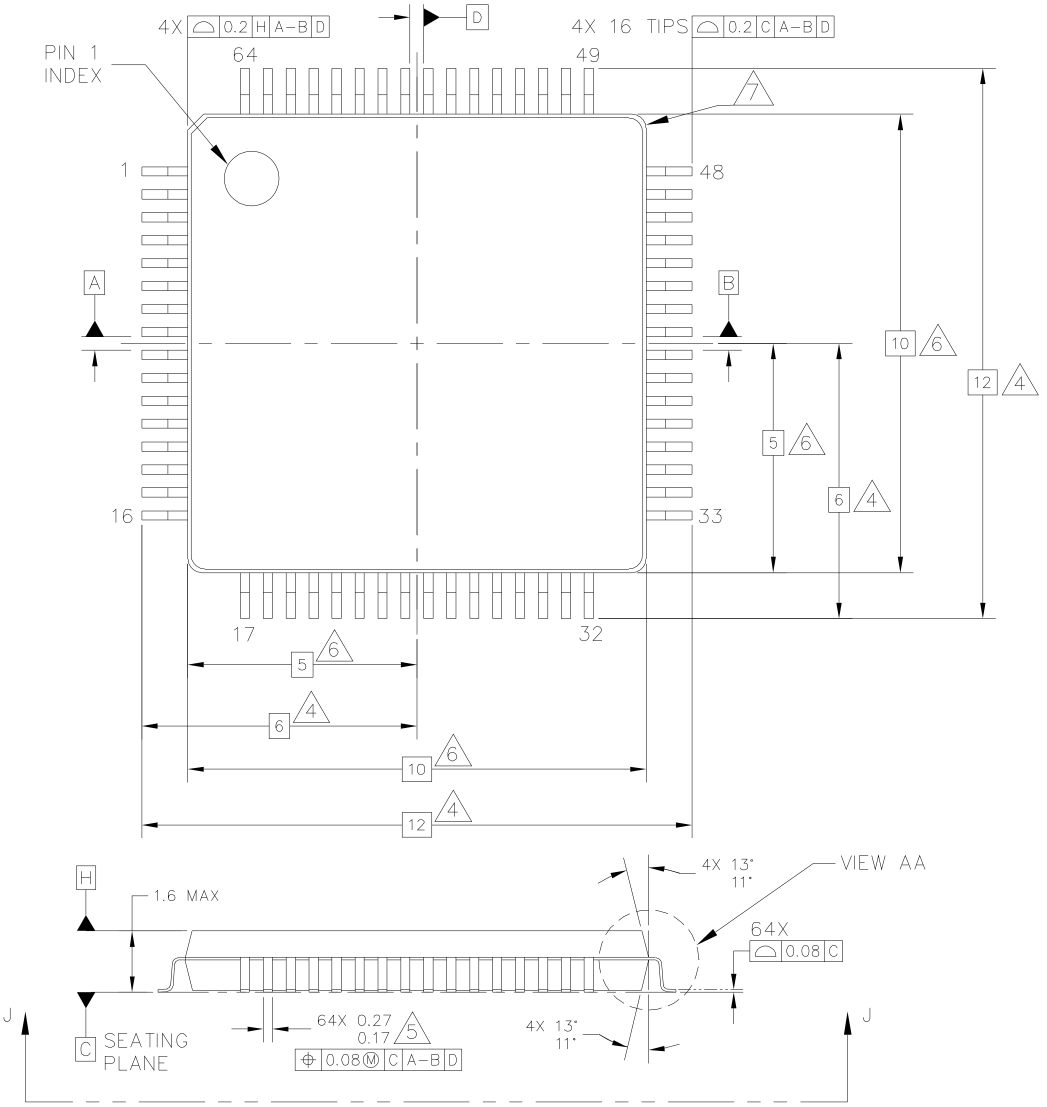
11 Packaging

11.1 Package Mechanical Dimensions

Package dimensions are provided in package drawings. To find the most current package outline drawing, go to www.freescale.com and perform a keyword search for the drawing's document number.

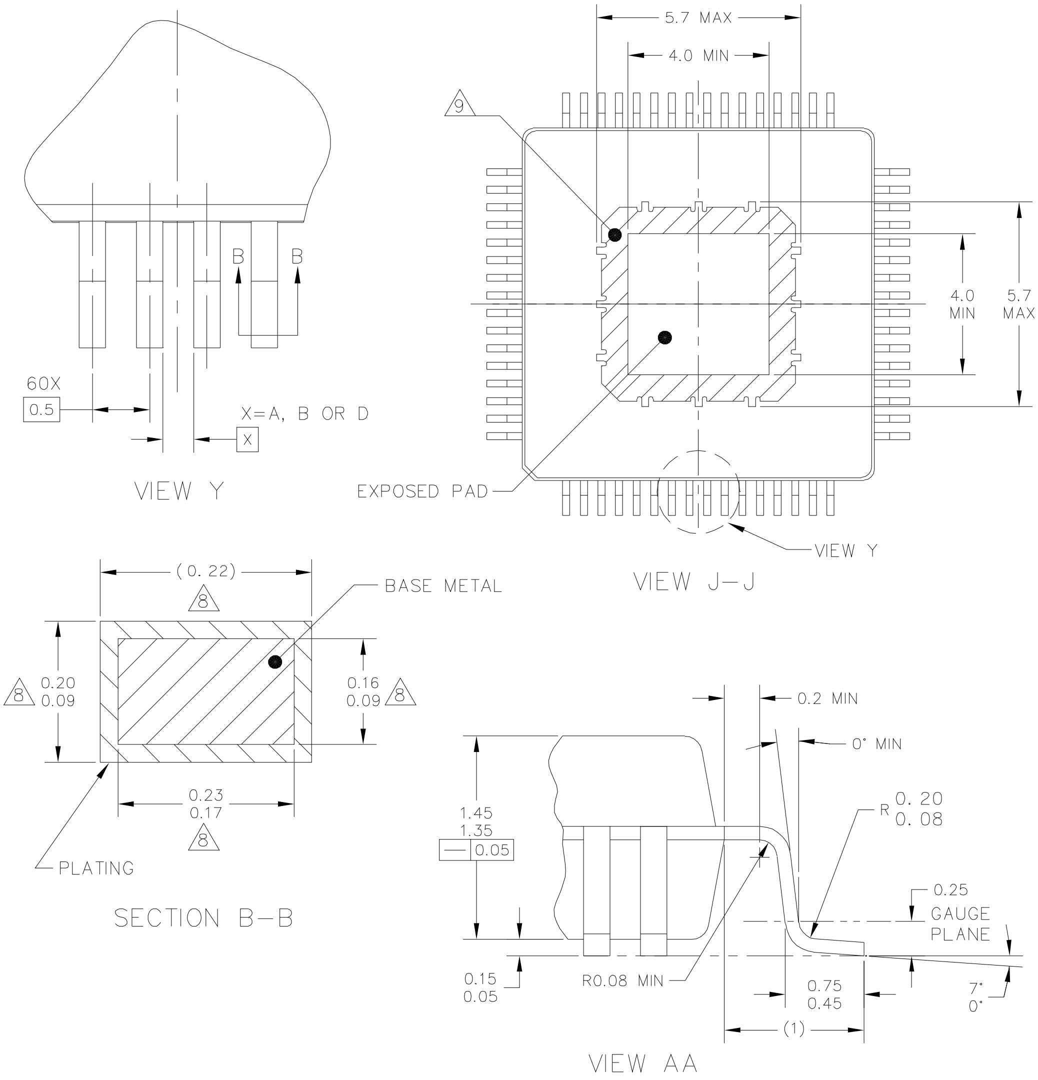
Table 230.

Package	Suffix	Package Outline Drawing Number
64-Pin LQFP Exposed Pad	AE	98ASA00237D



© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	**MECHANICAL OUTLINE**	PRINT VERSION NOT TO SCALE
TITLE: 64LD LQFP, 10 X 10 X 1.4 PKG, 0.5 PITCH, 4.9 X 4.9 EXPOSED PAD	DOCUMENT NO: 98ASA00237D	REV: O
	CASE NUMBER: 2139-01	01 MAR 2011
	STANDARD: JEDEC MS-026 BCD	

AE SUFFIX
64-PIN LQFP-EP
98ASA00237D
ISSUE 0



© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	**MECHANICAL OUTLINE**	PRINT VERSION NOT TO SCALE	
TITLE: 64LD LQFP, 10 X 10 X 1.4 PKG, 0.5 PITCH, 4.9 X 4.9 EXPOSED PAD	DOCUMENT NO: 98ASA00237D		REV: 0
	CASE NUMBER: 2139-01		01 MAR 2011
	STANDARD: JEDEC MS-026 BCD		

AE SUFFIX
64-PIN LQFP-EP
98ASA00237D
ISSUE 0

NOTES:

1. DIMENSIONS ARE IN MILLIMETERS.

2. DIMENSIONING AND TOLERANCING PER ASME Y14.5M-1994.

3. DATUMS A, B AND D TO BE DETERMINED AT DATUM PLANE H.

△4. DIMENSION TO BE DETERMINED AT SEATING PLANE C.

△5. THIS DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED THE UPPER LIMIT BY MORE THAN 0.08 MM AT MAXIMUM MATERIAL CONDITION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT. MINIMUM SPACE BETWEEN PROTRUSION AND ADJACENT LEAD SHALL NOT BE LESS THAN 0.07 MM.

△6. THIS DIMENSION DOES NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 MM PER SIDE. THIS DIMENSION IS MAXIMUM PLASTIC BODY SIZE DIMENSION INCLUDING MOLD MISMATCH.

△7. EXACT SHAPE OF EACH CORNER IS OPTIONAL.

△8. THESE DIMENSIONS APPLY TO THE FLAT SECTION OF THE LEAD BETWEEN 0.1 MM AND 0.25 MM FROM THE LEAD TIP.

△9. HATCHED AREA TO BE KEEP OUT ZONE FOR PCB ROUTING.

	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE
TITLE: 64LD LQFP, 10 X 10 X 1.4 PKG, 0.5 PITCH, 4.9 X 4.9 EXPOSED PAD	DOCUMENT NO: 98ASA00237D	REV: 0
	CASE NUMBER: 2139-01	01 MAR 2011
	STANDARD: JEDEC MS-026 BCD	

AE SUFFIX
64-PIN LQFP-EP
98ASA00237D
ISSUE 0

12 Revision History

REVISION	DATE	DESCRIPTION OF CHANGES
1.0	7/2012	• Initial release
2.0	11/2012	• Wording and part definitions changes were made. No electrical content was altered.
3.0	1/2014	• Corrected errors, typos, and formatting • Added HSX leakage current when pre-driver on (biasing switched off) and S_HSX leakage current delta between pre-drivers off and on parameters • Changed min for VB_HSX_VCCP__TH_R and VB_HSX_VCCP__TH_F • Added missing sentence for Cipher_unit Block
	1/2014	• Updated Data Sheet title
4.0	4/2014	• Redefined the minimum for t_{CSBF_SCLKR} and t_{SCLKF_CSBR} in Table 74. • Removed associated table and graphic for t_{CSBF_SCLKR} and t_{SCLKF_CSBR}.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. SMARTMOS is a trademark of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2014 Freescale Semiconductor, Inc.

Document Number: MC33816
Rev. 4.0
4/2014