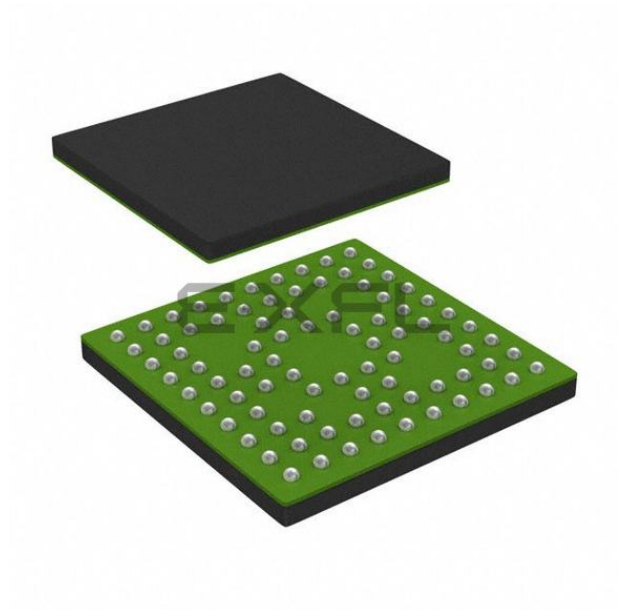


Welcome to E-XFL.COM

Embedded - Microcontrollers - Application Specific: Tailored Solutions for Precision and Performance

Embedded - Microcontrollers - Application Specific represents a category of microcontrollers designed with unique features and capabilities tailored to specific application needs. Unlike general-purpose microcontrollers, application-specific microcontrollers are optimized for particular tasks, offering enhanced performance, efficiency, and functionality to meet the demands of specialized applications.

What Are Embedded - Microcontrollers - Application Specific?

Application-specific microcontrollers are engineered to

Details	
Product Status	Active
Applications	Cryptography
Core Processor	ARM® Cortex®-M4F
Program Memory Type	-
Controller Series	-
RAM Size	480KB
Interface	I²C, SPI, UART
Number of I/O	65
Voltage - Supply	1.71V ~ 3.465V
Operating Temperature	0°C ~ 70°C
Mounting Type	Surface Mount
Package / Case	84-WFBGA
Supplier Device Package	84-WFBGA (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/cec1702q-b1-sx

Signal	Emulated Power Rail	Gated State	Notes
KSI1	VTR	Low	
KSI2	VTR	Low	
KSI3	VTR	Low	
KSI4	VTR	Low	
KSI5	VTR	Low	
KSI6	VTR	Low	
KSI7	VTR	Low	
KSO00	VTR	Low	
KSO01	VTR	Low	
KSO02	VTR	Low	
KSO03	VTR	Low	
KSO04	VTR	Low	
KSO05	VTR	Low	
KSO06	VTR	Low	
KSO07	VTR	Low	
KSO08	VTR	Low	
KSO09	VTR	Low	
KSO11	VTR	Low	
KSO12	VTR	Low	
LED0	VTR	Low	
LED1	VTR	Low	
PWM0	VTR	Low	
PWM1	VTR	Low	
PWM2	VTR	Low	
PWM3	VTR	Low	
PWM4	VTR	Low	
PWM5	VTR	Low	
PWM10	VTR	Low	
QSPI0_CLK	VTR	Low	
QSPI0_CS#	VTR	High	
QSPI0_IO0	VTR	Low	
QSPI0_IO1	VTR	Low	
QSPI0_IO2	VTR	Low	
QSPI0_IO3	VTR	Low	
QSPI1_CLK	VTR	Low	
QSPI1_CS#	VTR	High	
QSPI1_IO0	VTR	Low	
QSPI1_IO1	VTR	Low	
RC_ID1	VTR	Low	
RC_ID2	VTR	Low	
RESETI#	VTR	High	
SPI0_CLK	VTR	Low	
SPI0_CS#	VTR	High	

Feature	Instance	Base Address
Public Key Engine		4000_BD00h
Random Number Generator		4000_BE00h
Hash Engine		4000_D000h
Symmetric Encryption Engine		4000_D200h
Interrupt Aggregator		4000_E000h
EC Subsystem Registers		4000_FC00h
JTAG		4008_0000h
Power, Clocks and Resets		4008_0100h
GPIOs		4008_1000h
eFuse		4008_2000h
UART	0	400F_2400h
UART	1	400F_2800h
Real Time Clock		400F_5000h
Global Configuration		400F_FF00h

Agg IRQ	Agg Bit	HWB Instance Name	Interrupt Event	Wake Event	Source Description	Agg NVIC	Direct NVIC
GIRQ23	0	16-Bit Basic Timer 0	Timer_Event	No	Basic Timer Event	14	136
	1	16-Bit Basic Timer 1	Timer_Event	No	Basic Timer Event		137
	2	16-Bit Basic Timer 2	Timer_Event	No	Basic Timer Event		138
	3	16-Bit Basic Timer 3	Timer_Event	No	Basic Timer Event		139
	4	32-Bit Basic Timer 0	Timer_Event	No	Basic Timer Event		140
	5	32-Bit Basic Timer 1	Timer_Event	No	Basic Timer Event		141
	6	Counter/Timer 0	Timer_Event	No	16-bit Timer/Counter Event		142
	7	Counter/Timer 1	Timer_Event	No	16-bit Timer/Counter Event		143
	8	Counter/Timer 2	Timer_Event	No	16-bit Timer/Counter Event		144
	9	Counter/Timer 3	Timer_Event	No	16-bit Timer/Counter Event		145
	10	Capture Compare Timer	CAPTURE TIMER	No	CCT Counter Event		146
	11-13	Reserved					149
	14	Capture Compare Timer	CAPTURE 3	No	CCT Capture 3 Event		150
	15	Reserved					151
	16	Capture Compare Timer	CAPTURE 5	No	CCT Capture 5 Event		152
	17	Capture Compare Timer	COMPARE 0	No	CCT Compare 0 Event		153
	18-31	Reserved					

Block	Instance	Register	Register Address
GPIO	0	GPIO150 Pin Control 2 Register	400816A0h
GPIO	0	GPIO154 Pin Control 2 Register	400816B0h
GPIO	0	GPIO155 Pin Control 2 Register	400816B4h
GPIO	0	GPIO156 Pin Control 2 Register	400816B8h
GPIO	0	GPIO157 Pin Control 2 Register	400816BCh
GPIO	0	GPIO162 Pin Control 2 Register	400816C8h
GPIO	0	GPIO163 Pin Control 2 Register	400816CCh
GPIO	0	GPIO165 Pin Control 2 Register	400816D4h
GPIO	0	GPIO166 Pin Control 2 Register	400816D8h
GPIO	0	GPIO170 Pin Control 2 Register	400816E0h
GPIO	0	GPIO171 Pin Control 2 Register	400816E4h
GPIO	0	GPIO200 Pin Control 2 Register	40081700h
GPIO	0	GPIO201 Pin Control 2 Register	40081704h
GPIO	0	GPIO202 Pin Control 2 Register	40081708h
GPIO	0	GPIO203 Pin Control 2 Register	4008170Ch
GPIO	0	GPIO204 Pin Control 2 Register	40081710h
GPIO	0	GPIO223 Pin Control 2 Register	4008174Ch
GPIO	0	GPIO224 Pin Control 2 Register	40081750h
GPIO	0	GPIO225 Pin Control 2 Register	40081754h
GPIO	0	GPIO227 Pin Control 2 Register	4008175Ch
eFuse	0	Control Register	40082000h
eFuse	0	Manual Control Register	40082004h
eFuse	0	Manual Mode Address Register	40082006h
eFuse	0	Manual Mode Data Register	4008200Ch
eFuse	0	eFUSE Memory	40082010h
UART	0	Receive Buffer Register	400F2400h
UART	0	Transmit Buffer Register	400F2400h
UART	0	Programmable Baud Rate Generator LSB Register	400F2400h
UART	0	Programmable Baud Rate Generator MSB Register	400F2401h
UART	0	Interrupt Enable Register	400F2401h
UART	0	FIFO Control Register	400F2402h
UART	0	Interrupt Identification Register	400F2402h
UART	0	Line Control Register	400F2403h
UART	0	Modem Control Register	400F2404h
UART	0	Line Status Register	400F2405h
UART	0	Modem Status Register	400F2406h
UART	0	Scratchpad Register	400F2407h
UART	0	Activate Register	400F2730h
UART	0	Configuration Select Register	400F27F0h
UART	1	Receive Buffer Register	400F2800h
UART	1	Transmit Buffer Register	400F2800h

7.0 RAM AND ROM

7.1 References

None.

7.2 SRAM

The CEC1702 contains two blocks of SRAM. The two SRAM blocks in the CEC1702 total 480KB. Both SRAM blocks can be used for either program or data accesses. Performance is enhanced when program fetches and data accesses are to different SRAM blocks, but a program will operate correctly even if both program and data accesses are targeting the same block simultaneously.

- The first SRAM, which is optimized for code access, is 416KB
- The second SRAM, which is optimized for data access, is 64KB

7.3 ROM

The CEC1702 contains a 64KB block of ROM, located at address 00000000h in the ARM address space. The ROM contains boot code that is executed after the de-assertion of RESET_SYS. The boot code loads an executable code image into SRAM. The ROM also includes a set of API functions that can be used for cryptographic functions, as well as loading SRAM with programs or data.

7.4 Additional Memory Regions

7.4.1 ALIAS RAM

The Alias RAM region, starting at address 20000000h, is an alias of the SRAM located at 118000h, and is the same size as that SRAM block. EC software can access memory in either the primary address or in the alias region; however, access is considerably slower to the alias region. The alias region exists in order to enable the ARM bit-band region located at address 20000000h.

7.4.2 RAM BIT-BAND REGION

The RAM bit-band region is an alias of the SRAM located at 118000h, except that each bit is aliased to bit 0 of a 32-bit doubleword in the bit-band region. The upper 31 bits in each doubleword of the bit-band region are always 0. The bit-band region is therefore 32 times the size of the SRAM region. It can be used for atomic updates of individual bits of the SRAM, and is a feature of the ARM architecture.

The bit-band region can only be accessed by the ARM processor. Accesses by any other bus master will cause a memory fault.

7.4.3 CRYPTOGRAPHIC RAM

The cryptographic RAM is used by the cryptographic API functions in the ROM

7.4.4 REGISTER BIT-BAND REGION

The Register bit-band region is an 32-to-1 alias of the device register space starting at address 40000000h and ending with the Host register space at 400FFFFF. Every bit in the register space is aliased to a byte in the Register bit-band region, and like the RAM bit-band region, can be used by EC software to read and write individual register bits. Only the EC Device Registers and the GPIO Registers can be accessed via the bit-band region.

A one bit write operation to a register bit in the bit-band region is implemented by the ARM processor by performing a read, a bit modification, followed by a write back to the same register. Software must be careful when using bit-banding if a register contains bits have side effects triggered by a read.

The bit-band region can only be accessed by the ARM processor. Accesses by any other bus master will cause a memory fault.

7.5 Memory Map

The memory map of the RAM and ROM is represented as follows:

8.9.7 DMA CHANNEL N CONTROL REGISTER

Offset	10h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:26	Reserved	R	-	-
25	TRANSFER_ABORT This is used to abort the current transfer on this DMA Channel. The aborted transfer will be forced to terminate immediately.	R/W	0h	RESET
24	TRANSFER_GO This is used for the **Firmware Flow Control** DMA transfer. This is used to start a transfer under the **Firmware Flow Control**. Do not use this in conjunction with the **Hardware Flow Control**; DISABLE_HARDWARE_FLOW_CONTROL must be set in order for this field to function correctly.	R/W	0h	RESET
23	Reserved	R	-	-
22:20	TRANSFER_SIZE This is the transfer size in Bytes of each Data Packet transfer. The transfer size must be a legal transfer size. Valid sizes are 1, 2 and 4 Bytes.	R/W	0h	RESET
19	DISABLE_HARDWARE_FLOW_CONTROL Setting this bit to '1'b will Disable **Hardware Flow Control**. When disabled, any DMA Master device attempting to communicate to the DMA over the DMA Flow Control Interface will be ignored. This should be set before using the DMA channel in **Firmware Flow Control** mode.	R/W	0h	RESET
18	LOCK_CHANNEL This is used to lock the arbitration of the Channel Arbiter on this channel once this channel is granted. Once this is locked, it will remain on the arbiter until it has completed it transfer (either the Transfer Aborted, Transfer Done or Transfer Terminated conditions). **Note:** This setting may starve other channels if the locked channel takes an excessive period of time to complete.	R/W	0h	RESET
17	INCREMENT_DEVICE_ADDRESS If this bit is '1'b, the DEVICE_ADDRESS will be incremented by TRANSFER_SIZE after every Data Packet transfer	R/W	0h	RESET
16	INCREMENT_MEMORY_ADDRESS If this bit is '1'b, the MEMORY_START_ADDRESS will be incremented by TRANSFER_SIZE after every Data Packet transfer **Note:** If this is not set, the DMA will never terminate the transfer on its own. It will have to be terminated through the Hardware Flow Control or through a DMA Channel Control:Transfer Abort.	R/W	0h	RESET

9.5 Signal Description

9.5.1 SIGNAL INTERFACE

There are no external signals for this block.

9.6 Host Interface

The registers defined for the EC Interrupt Aggregator are only accessible by the embedded controller via the EC Registers.

9.7 Power, Clocks and Reset

9.7.1 BLOCK POWER DOMAIN

TABLE 9-1: BLOCK POWER

Power Well Source	Effect on Block
VTR	The EC Interrupt Aggregator block and registers operate on this single power well.

9.7.2 BLOCK CLOCKS

None

9.7.3 BLOCK RESET

TABLE 9-2: BLOCK RESETS

Reset Name	Reset Description
RESET_SYS	This signal is used to indicate when the VTR logic and registers in this block are reset.

9.8 Interrupts

This block aggregates all the interrupts targeted for the embedded controller into the Source Registers defined in Section 9.11, "EC Registers". The unmasked bits of each source register are then OR'd together and routed to the embedded controller's interrupt interface. The name of each Source Register identifies the IRQ number of the interrupt port on the embedded controller.

9.9 Low Power Modes

This block always automatically adjusts to operate in the lowest power mode.

9.10 Description

The interrupt generation logic is made of groups of signals, each of which consist of a Status register, a Enable Set register, and Enable Clear register and a Result register.

The Status and Enable are latched registers. There is one set of Enable register bits; both the Enable Set and Enable Clear registers return the same result when read. The Enable Set interface is used to set individual bits in the Enable register, and the Enable Clear is used to clear individual bits. The Result register is a bit by bit AND function of the Source and Enable registers. All the bits of the Result register are OR'ed together and AND'ed with the corresponding bit in the Block Select register to form the interrupt signal that is routed to the ARM interrupt controller.

The Result register bits may also be enabled to the NVIC block via the NVIC_EN bit in the Interrupt Control Register register. See Chapter 34.0, "EC Subsystem Registers"

Section 9.10.1 shows a representation of the interrupt structure.

TABLE 11-6: EDGE ENABLE AND INTERRUPT DETECTION BITS DEFINITION

Edge Enable	Interrupt Detection Bits			Selected Function
D7	D6	D5	D4	
0	0	0	0	Low Level Sensitive
0	0	0	1	High Level Sensitive
0	0	1	0	Reserved
0	0	1	1	Reserved
0	1	0	0	Interrupt events are disabled
0	1	0	1	Reserved
0	1	1	0	Reserved
0	1	1	1	Reserved
1	1	0	1	Rising Edge Triggered
1	1	1	0	Falling Edge Triggered
1	1	1	1	Either edge triggered

Note: Only edge triggered interrupts can wake up the main clock domain. The GPIO must be enabled for edge-triggered interrupts and the GPIO interrupt must be enabled in the interrupt aggregator in order to wake from the Heavy Sleep state.

11.10.2 PIN CONTROL 2 REGISTER

Offset	See Section 11.8			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:6	Reserved	R	-	-
5:4	DRIVE_STRENGTH These bits are used to select the drive strength on the pin. The drive strength is the same whether the pin is powered by 3.3V or 1.8V. 11b=12mA 10b=8mA 01b=4mA 00b=2mA	R/W	See Section 11.8	RESET_SYS
3:1	Reserved	R	-	-
0	SLEW_RATE This bit is used to select the slew rate on the pin. 1=fast 0=slow (half frequency)	R/W	0h	RESET_SYS

11.10.3 GPIO INPUT REGISTERS

The GPIO Input Registers can always be used to read the state of a pin, even when the pin is configured as an output, /or when the pin is configured for a signal function other than the GPIO (i.e., the MUX_CONTROL field is not equal to '00.').

Note: Bits associated with GPIOs not present in the pinout for a particular device are Reserved.

In typical systems, the fans are powered by the main power supply. Firmware may disable this block when it detects that the main power rail has been turned off by either clearing the <enable> TACH_ENABLE bit or putting the block to sleep via the supported Low Power Mode interface (see Low Power Modes).

20.10.1 MODES OF OPERATION

The Tachometer block supports two modes of operation. The mode of operation is selected via the TACH_READING_-MODE_SELECT bit.

20.10.1.1 Free Running Counter

In Mode 0, the Tachometer block uses the TACH input as the clock source for the internal TACH pulse counter (see TACHX_COUNTER). The counter is incremented when it detects a rising edge on the TACH input. In this mode, the firmware may periodically poll the TACHX_COUNTER field to determine the average speed over a period of time. The firmware must store the previous reading and the current reading to compute the number of pulses detected over a period of time. In this mode, the counter continuously increments until it reaches FFFFh. It then wraps back to 0000h and continues counting. The firmware must ensure that the sample rate is greater than the time it takes for the counter to wrap back to the starting point.

Note:	Tach interrupts should be disabled in Mode 0.

20.10.1.2 Mode 1 -- Number of Clock Pulses per Revolution

In Mode 1, the Tachometer block uses its 100KHz clock input to measure the programmable number of TACH pulses. In this mode, the internal TACH pulse counter (TACHX_COUNTER) returns the value in number of 100KHz pulses per programmed number of TACH_EDGES. For fans that generate two square waves per revolution, these bits should be configured to five edges.

When the number of edges is detected, the counter is latched and the COUNT_READY_STATUS bit is asserted. If the COUNT_READY_INT_EN bit is set a TACH interrupt event will be generated.

20.10.2 OUT-OF-LIMIT EVENTS

The TACH Block has a pair of limit registers that may be configured to generate an event if the Tach indicates that the fan is operating too slow or too fast. If the <TACH reading> exceeds one of the programmed limits, the TACHx High Limit Register and the TACHx Low Limit Register, the bit TACH_OUT_OF_LIMIT_STATUS will be set. If the TACH_OUT_OF_LIMIT_STATUS bit is set, the Tachometer block will generate an interrupt event.

20.11 EC Registers

Registers for this block are shown in the following summary table. Addresses for each register are determined by adding the offset to the Base Address for each instance of the TACH Block in the Block Overview and Base Address Table in Section 3.0, "Device Inventory".

TABLE 20-3: REGISTER SUMMARY

Offset	Register Name
00h	TACHx Control Register
04h	TACHx Status Register
08h	TACHx High Limit Register
0Ch	TACHx Low Limit Register

21.11.2 PWMX COUNTER OFF TIME REGISTER

Offset	04h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:16	Reserved	R	-	-
15:0	PWMX_COUNTER_OFF_TIME This field determine both the frequency and duty cycle of the PWM signal. Setting this field to a value of *n* will cause the Off time of the PWM to be *n*+1 cycles of the PWM Clock Source. When this field is set to zero, the PWM_OUTPUT is held high (Full On).	R/W	FFFFh	RESET_SYS

21.11.3 PWMX CONFIGURATION REGISTER

Offset	08h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:7	Reserved	R	-	-
6:3	CLOCK_PRE_DIVIDER The Clock source for the 16-bit down counter (see PWMx Counter ON Time Register and PWMx Counter OFF Time Register) is determined by bit D1 of this register. The Clock source is then divided by the value of Pre-Divider+1 and the resulting signal determines the rate at which the down counter will be decremented. For example, a Pre-Divider value of 1 divides the input clock by 2 and a value of 2 divides the input clock by 3. A Pre-Divider of 0 will disable the Pre-Divider option.	R/W	0000b	RESET_SYS
2	INVERT 1=PWM_OUTPUT ON State is active low 0=PWM_OUTPUT ON State is active high	R/W	0b	RESET_SYS
1	CLOCK_SELECT This bit determines the clock source used by the PWM duty cycle and frequency control logic. 1=CLOCK_LOW 0=CLOCK_HIGH	R/W	0b	RESET_SYS
0	PWM_ENABLE When the PWM_ENABLE is set to 0 the internal counters are reset and the internal state machine is set to the OFF state. In addition, the PWM_OUTPUT signal is set to the inactive state as determined by the Invert bit. The PWMx Counter ON Time Register and PWMx Counter OFF Time Register are not affected by the PWM_ENABLE bit and may be read and written while the PWM enable bit is 0. 1=Enabled (default) 0=Disabled (gates clocks to save power)	R/W	0b	RESET_SYS

24.0 BLINKING/BREATHING PWM

24.1 Introduction

LEDs are used in computer applications to communicate internal state information to a user through a minimal interface. Typical applications will cause an LED to blink at different rates to convey different state information. For example, an LED could be full on, full off, blinking at a rate of once a second, or blinking at a rate of once every four seconds, in order to communicate four different states.

As an alternative to blinking, an LED can “breathe”, that is, oscillate between a bright state and a dim state in a continuous, or apparently continuous manner. The rate of breathing, or the level of brightness at the extremes of the oscillation period, can be used to convey state information to the user that may be more informative, or at least more novel, than traditional blinking.

The blinking/breathing hardware is implemented using a PWM. The PWM can be driven either by the Main system clock or by a 32.768 KHz clock input. When driven by the Main system clock, the PWM can be used as a standard 8-bit PWM in order to control a fan. When used to drive blinking or breathing LEDs, the 32.768 KHz clock source is used.

Features:

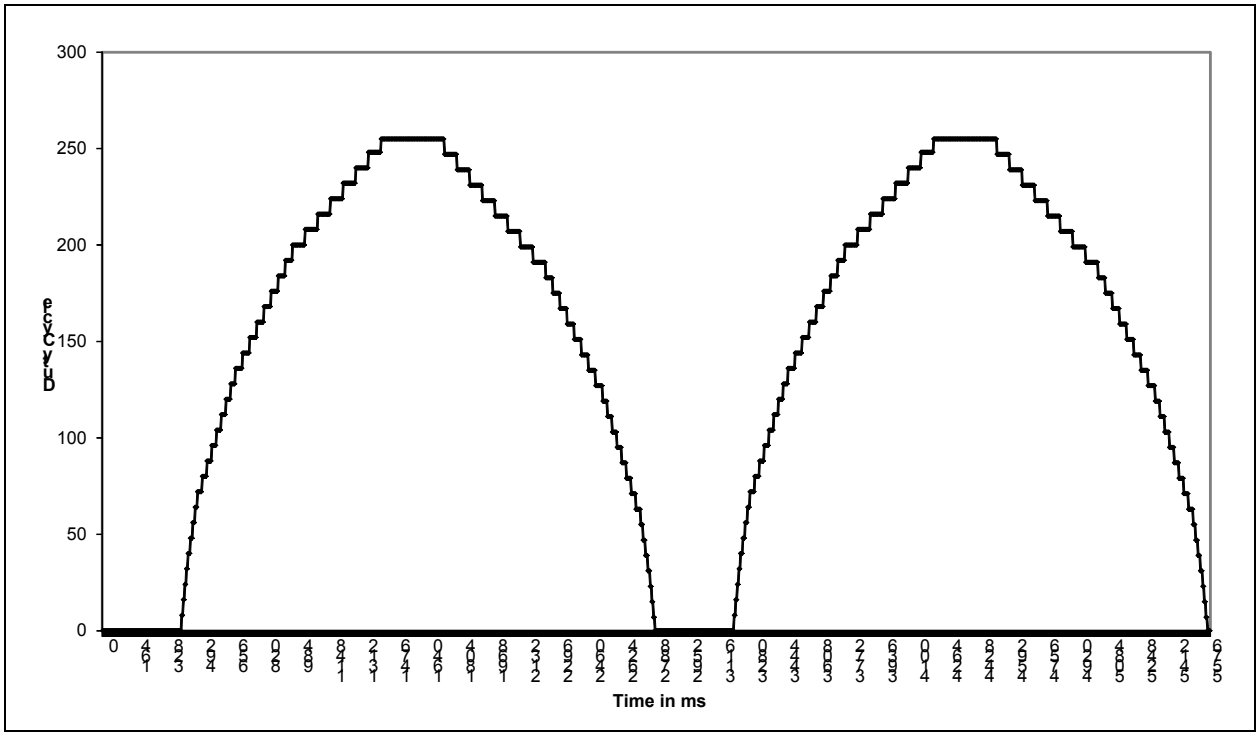
- Each PWM independently configurable
- Each PWM configurable for LED blinking and breathing output
- Highly configurable breathing rate from 60ms to 1min
- Non-linear brightness curves approximated with 8 piece wise-linear segments
- All LED PWMs can be synchronized
- Each PWM configurable for 8-bit PWM support
- Multiple clock rates
- Configurable Watchdog Timer

24.2 Interface

This block is designed to drive a pin on the pin interface and to be accessed internally via a registered host interface.

The resulting curve is shown in the following figure:

FIGURE 24-6: NON-LINEAR BRIGHTNESS CURVE EXAMPLE



24.10 EC Registers

Registers for this block are shown in the following summary table. Addresses for each register are determined by adding the offset to the Base Address for each instance of the Blinking/Breathing PWM Block in the Block Overview and Base Address Table in Section 3.0, "Device Inventory".

TABLE 24-10: REGISTER SUMMARY

Offset	Register Name
00h	LED Configuration Register
04h	LED Limits Register
08h	LED Delay Register
0Ch	LED Update Stepsize Register
10h	LED Update Interval Register
14h	LED Output Delay

In the following register definitions, a “PWM period” is defined by time the PWM counter goes from 000h to its maximum value (FFh in 8-bit mode, FEh in 7-bit mode and FCh in 6-bit mode, as defined by the PSCALE field in register LED_CFG). The end of a PWM period occurs when the PWM counter wraps from its maximum value to 0.

The registers in this block can be written 32-bits, 16-bits or 8-bits at a time. Writes to LED Configuration Register take effect immediately. Writes to LED Limits Register are held in a holding register and only take effect only at the end of a PWM period. The update takes place at the end of every period, even if only one byte of the register was updated. This means that in blink/PWM mode, software can change the duty cycle with a single 8-bit write to the MIN field in the LED_LIMIT register. Writes to LED Delay Register, LED Update Stepsize Register and LED Update Interval Register also go initially into a holding register. The holding registers are copied to the operating registers at the end of a PWM period only if the Enable Update bit in the LED Configuration Register is set to 1. If LED_CFG is 0, data in the holding registers is retained but not copied to the operating registers when the PWM period expires. To change an LED breath-

26.0 KEYBOARD SCAN INTERFACE

26.1 Overview

The Keyboard Scan Interface block provides a register interface to the EC to directly scan an external keyboard matrix of size up to 18x8.

The maximum configuration of the Keyboard Scan Interface is 18 outputs by 8 inputs. For a smaller matrix size, firmware should configure unused KSO pins as GPIOs or another alternate function, and it should mask out unused KSIs and associated interrupts.

26.2 References

No references have been cited for this feature.

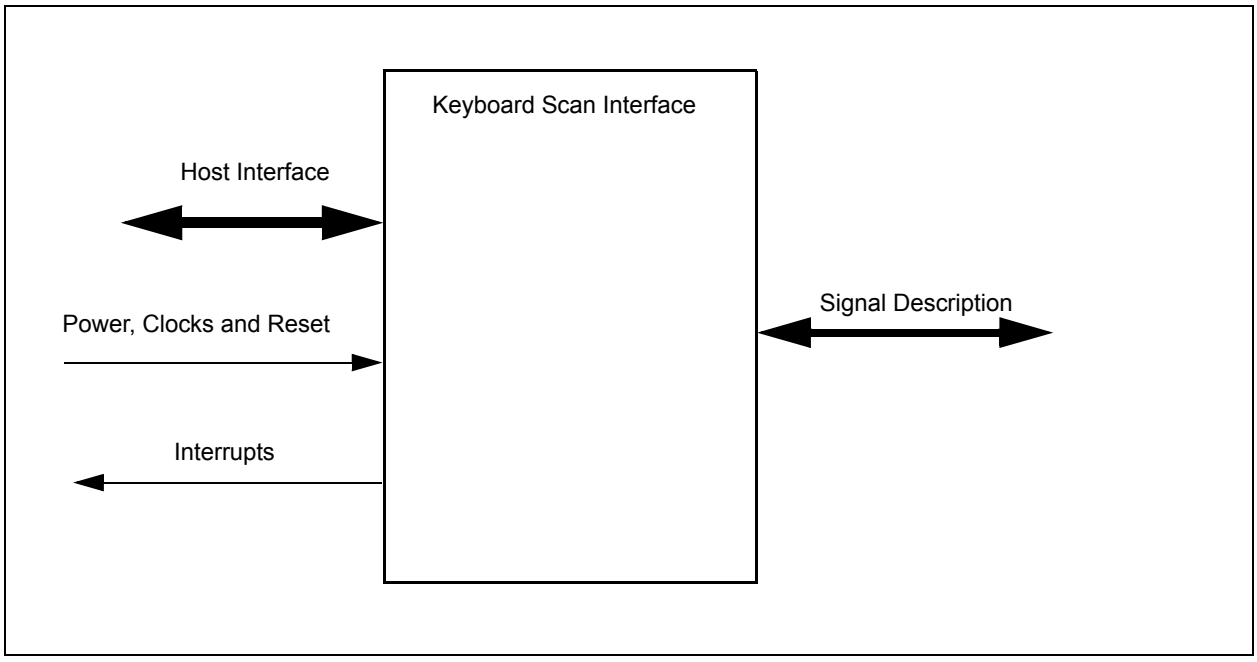
26.3 Terminology

There is no terminology defined for this section.

26.4 Interface

This block is designed to be accessed externally via the pin interface and internally via a registered host interface.

FIGURE 26-1: I/O DIAGRAM OF BLOCK



26.5 Signal Description

Name	Direction	Description
KSI[7:0]	Input	Column inputs from external keyboard matrix.
KSO[17:0]	Output	Row outputs to external keyboard matrix.

26.11.4 KSI INTERRUPT ENABLE REGISTER

Offset	10h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:8	Reserved	R	-	-
7:0	KSI_INT_EN Each bit in KSI_INT_EN enables interrupt generation due to high-to-low transition on a KSI input. An interrupt is generated when the corresponding bits in KSI_STATUS and KSI_INT_EN are both set.	R/W	0h	RESET_SYS

26.11.5 KEYSCAN EXTENDED CONTROL REGISTER

Offset	14h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
32:1	Reserved	R	-	-
0	PREDRIVE_ENABLE PREDRIVE_ENABLE enables the PREDRIVE mode to actively drive the KSO pins high for two 48 MHz PLL clocks before switching to open-drain operation. 0=Disable predrive on KSO pins 1=Enable predrive on KSO pins.	R/W	0h	RESET_SYS

bus bit by bit, the Tx Empty signal is again asserted, triggering the DMA fetch-and-write cycle. The process continues until the end of the DMA buffer is reached - the DMA controller stops responding to an active Tx Empty until the buffer's address registers are reprogrammed.

SPI receive / DMA read: the AUTO_READ bit in the SPI Control Register must be set. The driver first writes (dummy data) to the SPI TX Data Register (SPITD) to initiate the toggling of the SPI clock, enabling data to be shifted in. After one byte is received, the Rx Full (RxBF) status signal, used as a read request to the DMA controller, is asserted. The DMA controller then reads the received byte from the GP-SPI's SPI RX Data Register (SPIRD) and stores it in the DMA receive buffer. With AUTO_READ set, this read clears both the RxBF and TxBE. Clearing TxBE causes (dummy) data from the SPI TX Data Register (SPITD) to be transferred to the internal shift register, mimicking the effect of the aforementioned write to the SPI TX Data Register (SPITD) by the driver. SPI clock is toggled again to shift in the second read byte. This process continues until the end of the DMA buffer is reached - the DMA controller stops responding to an active Tx Empty until the buffer's address registers are reprogrammed.

28.10.3 TYPES OF SPI TRANSACTIONS

The GP-SPI controller can be configured to operate in three modes: Full Duplex, Half Duplex, and Dual Mode.

28.10.3.1 Full Duplex

In Full Duplex Mode, serial data is transmitted and received simultaneously by the SPI master over the SPDOUT and SPDIN pins. To enable Full Duplex Mode clear SPDIN Select.

When a transaction is completed in the full-duplex mode, the RX_DATA shift register always contains received data (valid or not) from the last transaction.

28.10.3.2 Half Duplex

In Half Duplex Mode, serial data is transmitted and received sequentially over a single data line (referred to as the SPDOUT pin). To enable Half Duplex Mode set SPDIN Select to 01b. The direction of the SPDOUT signal is determined by the BIOEN bit.

- To transmit data in half duplex mode set the BIOEN bit before writing the TX_DATA register.
- To receive data in half duplex mode clear the BIOEN bit before writing the TX_DATA register with a dummy byte.

Note:	The Software driver must properly drive the BIOEN bit and store received data depending on the transaction format of the specific slave device.

28.10.3.3 Dual Mode of Operation

In Dual Mode, serial data is transmitted sequentially from the SPDOUT pin and received in by the SPI master from the SPDOUT and SPDIN pins. This essentially doubles the received data rate and is often available in SPI Flash devices. To enable Dual Mode of operation the SPI core must be configured to receive data in path on the SPDIN1 and SPDIN2 inputs via SPDIN Select. The BIOEN bit determines if the SPI core is transmitting or receiving. The setting of this bit determines the direction of the SPDOUT signal. The SPDIN Select bits are configuration bits that remain static for the duration of a dual read command. The BIOEN bit must be toggled to indicate when the SPI core is transmitting and receiving.

- To transmit data in dual mode set the BIOEN bit before writing the TX_DATA register.
- To receive data in dual mode clear the BIOEN bit before writing the TX_DATA register with a dummy byte. The even bits (0,2,4,and 6) are received on the SPDOUT pin and the odd bits (1,3,5,and 7) are received on the SPDIN pin. The hardware assembles these received bits into a single byte and loads them into the RX_DATA register accordingly.

29.11.2 QMSPI CONTROL REGISTER

Offset	04h			
Bits	**Description**	**Type**	**Default**	**Reset Event**
31:17	TRANSFER_LENGTH The length of the SPI transfer. The count is in bytes or bits, depending on the value of TRANSFER_LENGTH_BITS. A value of '0' means an infinite length transfer.	R/W	0h	RESET
16	DESCRIPTION_BUFFER_ENABLE This enables the Description Buffers to be used. 1=Description Buffers in use. The first buffer is defined in DESCRIPTION_BUFFER_POINTER 0=Description Buffers disabled	R/W	0h	RESET
15:12	DESCRIPTION_BUFFER_POINTER This field selects the first buffer used if Description Buffers are enabled.	R/W	0h	RESET
11:10	TRANSFER_UNITS 3=TRANSFER_LENGTH defined in units of 16-byte segments 2=TRANSFER_LENGTH defined in units of 4-byte segments 1=TRANSFER_LENGTH defined in units of bytes 0=TRANSFER_LENGTH defined in units of bits	R/W	0h	RESET
9	CLOSE_TRANSFER_ENABLE This selects what action is taken at the end of a transfer. When the transaction closes, the Chip Select de-asserts, the SPI interface returns to IDLE and the DMA interface terminates When Description Buffers are in use this bit must be set only on the Last Buffer. 1=The transaction is terminated 0=The transaction is not terminated	R/W	1h	RESET
8:7	RX_DMA_ENABLE This bit enables DMA support for Receive Transfer. If enabled, DMA will be requested to empty the FIFO until either the interface reaches TRANSFER_LENGTH or the DMA sends a termination request. The size defined here must match DMA programmed access size. 1=DMA is enabled.and set to 1 Byte 2=DMA is enabled and set to 2 Bytes 3=DMA is enabled and set to 4 Bytes 0=DMA is disabled. All data in the Receive Buffer must be emptied by firmware	R/W	0h	RESET
6	RX_TRANSFER_ENABLE This bit enables the receive function of the SPI interface. 1=Receive is enabled. Data received from the SPI Slave is stored in the Receive Buffer 0=Receive is disabled	R/W	0h	RESET

32.7 Power, Clocks and Reset

This section defines the Power, Clock, and Reset parameters of the block.

32.7.1 POWER DOMAINS

Name	Description
VTR	The main power well used when the VBAT RAM is accessed by the EC.
VBAT	The power well used to retain memory state while the main power rail is unpowered.

32.7.2 CLOCK INPUTS

No special clocks are required for this block.

32.7.3 RESETS

Name	Description
RESET_VBAT	This signal resets all the registers and logic in this block to their default state.

32.8 Interrupts

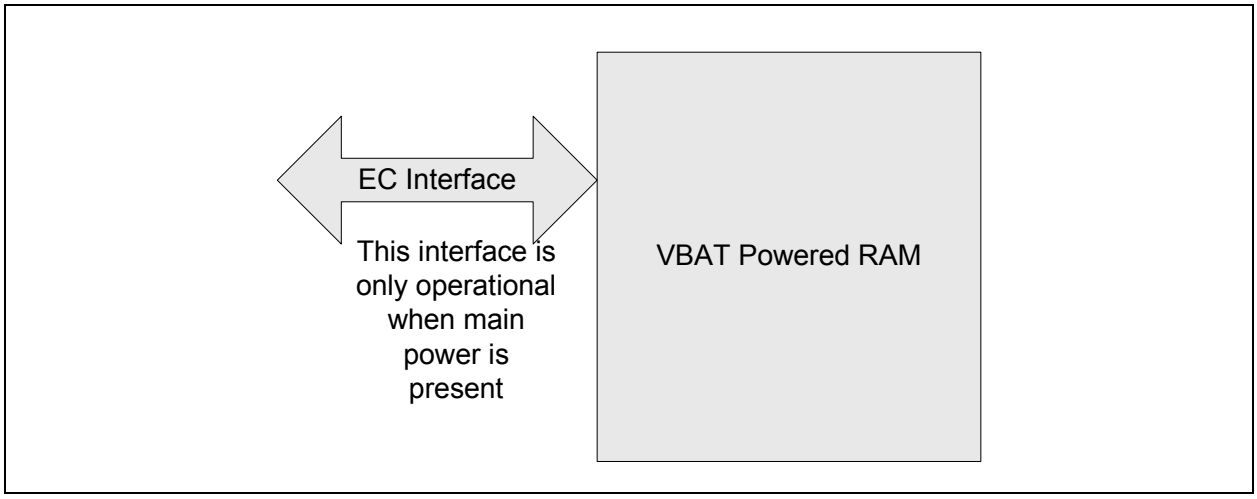
This block does not generate any interrupts.

32.9 Low Power Modes

The VBAT-Powered RAM automatically enters a low power mode whenever it is not being accessed by the EC. There is no chip-level Sleep Enable input.

32.10 Description

FIGURE 32-2: VBAT RAM BLOCK DIAGRAM



The VBAT Powered RAM provides a 128 Byte Random Accessed Memory that is operational while VTR is powered, and will retain its values powered by VBAT while VTR is unpowered. The RAM is organized as a 32 words x 32-bit wide for a total of 128 bytes.

The contents of the VBAT RAM is indeterminate after a RESET_VBAT.

36.0 TEST MECHANISMS

36.1 ARM Test Functions

Test mechanisms for the ARM are described in Section 6.0, "ARM M4F Based Embedded Controller".

36.2 JTAG Boundary Scan

Note:	Boundary Scan operates in 4-wire JTAG mode only. This is not supported by 2-wire SWD.

JTAG Boundary Scan includes registers and functionality as defined in IEEE 1149.1 and the CEC1702 BSDL file. Functionality implemented beyond the standard definition is summarized in Table 36-1. The CEC1702 Boundary Scan JTAG ID is shown in Table 1-1.

Note:	Must wait a minimum of 35ms after a POR to accurately read the Boundary Scan JTAG ID. Reading the JTAG ID too soon may return a Boundary Scan JTAG ID of 00000000h. This is not a valid ID value.

36.2.1 TAP CONTROLLER SELECT STRAP OPTION

The TAP Controller Select Strap Option determines the JTAG slave that is selected when JTAG_RST# is not asserted. The state of the TAP Controller Select Strap Option pin, defined in the Pin Configuration chapter, is sampled by hardware at POR according to the Slave Select Timing as defined in Section 39.16, "JTAG Interface Timing" and is registered internally to select between the debug and boundary scan TAP controllers.

If the strap is sampled low, the debug TAP controller is selected; if the strap is sampled high, the boundary scan slave is selected. An internal pull-up resistor is enabled by default on the TAP Controller Select Strap Option pin and can be disabled by firmware, if necessary.

TABLE 36-1: EXTENDED BOUNDARY SCAN FUNCTIONALITY

Bits	Function	Description
12, 14	TAP Controller Select Strap Option Override	When the Strap Option Override is ‘1,’ the strap option is overridden to select the debug TAP Controller until the next time the strap is sampled. To set Strap Override Function, write 0X1FFFFD to the TAP controller instruction register, then write 0x5000 to the TAP controller data register. Note that the instruction register is 18 bits long; the data register is 16 bits long.

36.3 JTAG Master

The JTAG Master controller in the CEC1702 enables the embedded controller to perform full IEEE 1149.1 test functions as the master controller for test operations at assembly time or in the field.

The JTAG Master interface shares the JTAG pin interface with the JTAG Boundary Scan and Debug TAP controllers; including, JTAG_CLK, JTAG_TDI, JTAG_TDO and JTAG_TMS. When the CEC1702 JTAG interface is configured as master, it is the responsibility of the master firmware to satisfy all requirements regarding JTAG port multiplexing. It is also it is the responsibility of the JTAG Master firmware to satisfy all requirements for external JTAG slave devices that require an external asynchronous reset (TRST#) input.

When JTAG slave functions are not required and the JTAG Master is enabled, the JTAG Interface pins are turned around so that the pins JTAG_CLK, JTAG_TMS and JTAG_TDI become outputs and the JTAG_TDO becomes an input.

Figure 36-1, "JTAG Signal Clocking" shows the clocking behavior of JTAG in the TAP controller in a JTAG Slave device. The rows “TAP State” and “Shift Reg. Contents” refer to the state of the JTAG Slave device and are provided for reference. When configured as a Master, the JTAG interface drives JTAG_CLK and will shift out data onto JTAG_TMS and JTAG_TDI in parallel, updating the pins on the falling edge of JTAG_CLK. The Master will sample data on JTAG_TDO on the rising edge of JTAG_CLK.

37.4 Power, Clocks and Resets

This section defines the Power, Clock, and Reset parameters of the block.

37.4.1 POWER DOMAINS

TABLE 37-2: POWER SOURCES

Name	Description
VTR	This power well sources all of the registers and logic in this block, except where noted.

37.4.2 CLOCKS

This section describes all the clocks in the block, including those that are derived from the I/O Interface as well as the ones that are derived or generated internally.

TABLE 37-3: CLOCKS

Name	Description
48MHz	This clock signal drives selected logic (e.g., counters).

37.4.3 RESETS

TABLE 37-4: RESET SIGNALS

Name	Description
RESET_SYS	This reset signal resets all of the registers and logic in this block.

37.5 Interrupt Generation

There are no interrupts from this block.

37.6 Low Power Modes

The eFUSE Block may be put into a low power state by the chip's Power, Clocks, and Reset (PCR) circuitry.

37.7 Description

The eFuse memory consists of four blocks of 1K bits each, for a total of 4K bits. The assignment of the eFuse data is shown in Section 37.8, "eFuse Memory Map".The eFUSE bits are programmed one bit at a time through a register interface. Addressing bits for writing bits is by a 2-bit block address field and a 10-bit offset within block field. The eFUSE memory can be read through 8-bit or 16-bit reads of a block of register addresses.

Note:	Only 8-bit and 16-bit access to the eFUSE memory is supported. A 32-bit access or larger will just have the 16-bit read-back value replicated in the other byte lanes.

37.7.1 EFUSE PROGRAMMING SEQUENCE

Programming of the eFuse array is one bit at a time. Programming changes a bit of '0b' to '1b'. There is no way to change the value of a bit that is already '1b'.

Sequence for programming one bit of eFuse memory:

1. Set the VREF_ADC pin to ground before powering up.
2. Power up the rest of the supplies
3. Set the MAN_ENABLE bit in the Manual Control Register to '1b'
4. Set the FSOURCE_EN_PRGM to '1b'
5. Set FSOURCE_EN_READ to '0b' - DO NOT combine with step 4; writing FSOURCE_EN_PRGM and FSOURCE_EN_READ MUST be performed with separate writes
6. Set the VREF_ADC pin to the VPP programming voltage of 1.59V.

39.2 Power Sequencing

FIGURE 39-3: POWER RAIL SEQUENCING

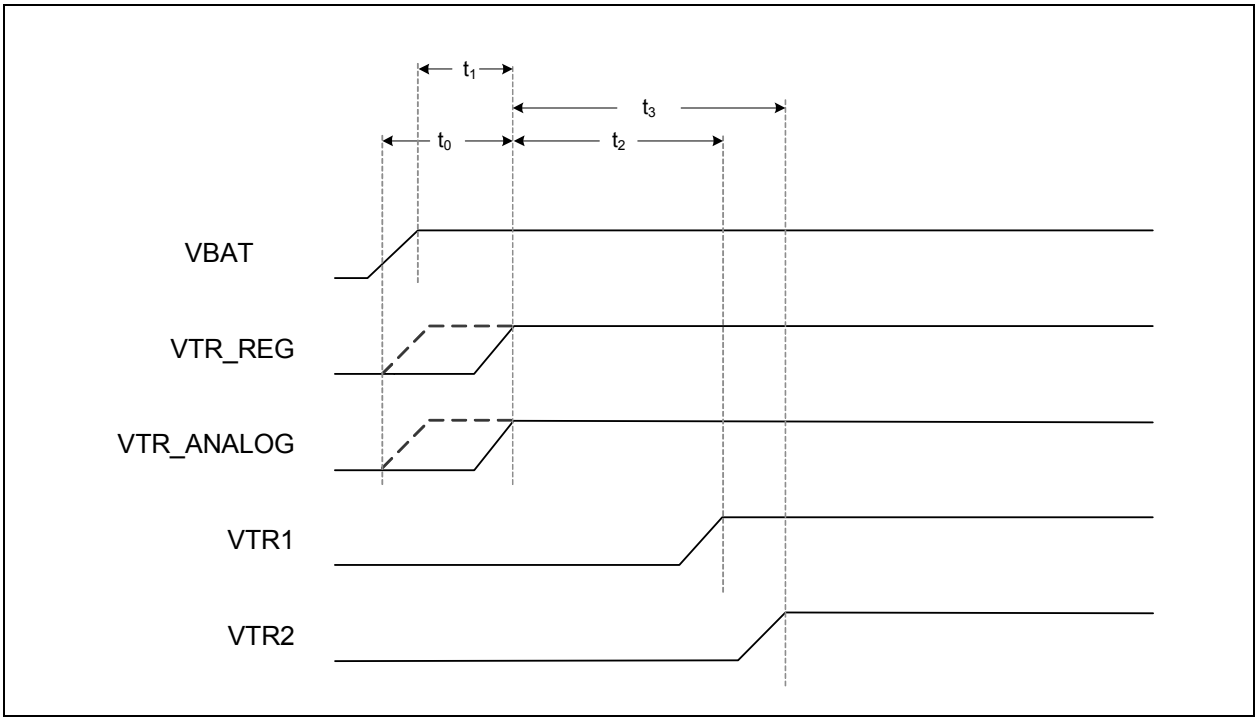


TABLE 39-2: POWER SEQUENCING PARAMETERS

Symbol	Parameter	Min	Typ	Max	Units	Notes
t_0	VTR_ANALOG stable relative to VTR_REG stable	0			ms	1
	VTR_REG stable relative to VTR_ANALOG stable	0			ms	
t_1	VBAT stable to VTR_ANALOG and VTR_REG stable	0			ms	2
t_2	VTR_ANALOG and VTR_REG stable to VTR1 stable, VTR1 at 3.3V	0		500	µs	
	VTR_ANALOG and VTR_REG stable to VTR1 stable, VTR1 at 1.8V	0		3	ms	

Note 1: VTR_ANALOG and VTR_REG may ramp in either order. There is no limit on the time between the ramp of one rail and the ramp of the other.

2: VBAT must rise no later than VTR_ANALOG and VTR_REG. This relationship is ensured by the recommended battery circuit.