Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

AN220222

Low Power Mode Procedure in Traveo II Family

Author: Wafa Syakira Binti Azmi

Associated Part Family: Traveo™ II Family CYT2/CYT3/CYT4 Series

Related Application Notes: see Related Documents

This application note describes the features of low-power modes in Traveo™ II family MCUs and explains how to enter low-power modes and return to active mode.

Contents

1 Introduction

This application note describes low-power modes in Cypress Traveo II family MCU. The series includes Arm® Cortex® CPUs, CAN FD, memory, and analog and digital peripheral functions in a single chip.

The CYT2 series has one Arm Cortex-M4F-based CPU (CM4) and Cortex-M0+-based CPU (CM0+). The CYT4 series has two Arm Cortex-M7-based CPUs (CM7) and CM0+, and the CYT3 series has one CM7 and CM0+.

Traveo II family MCUs have several different power modes. These modes are intended to minimize the average power consumption in an application.

This application note explains the features of power modes and how to set up the power mode transition.

To understand the described functionality and terminology used in this application note, see the “Device Power Modes” chapter of the Architecture Technical Reference Manual (TRM).

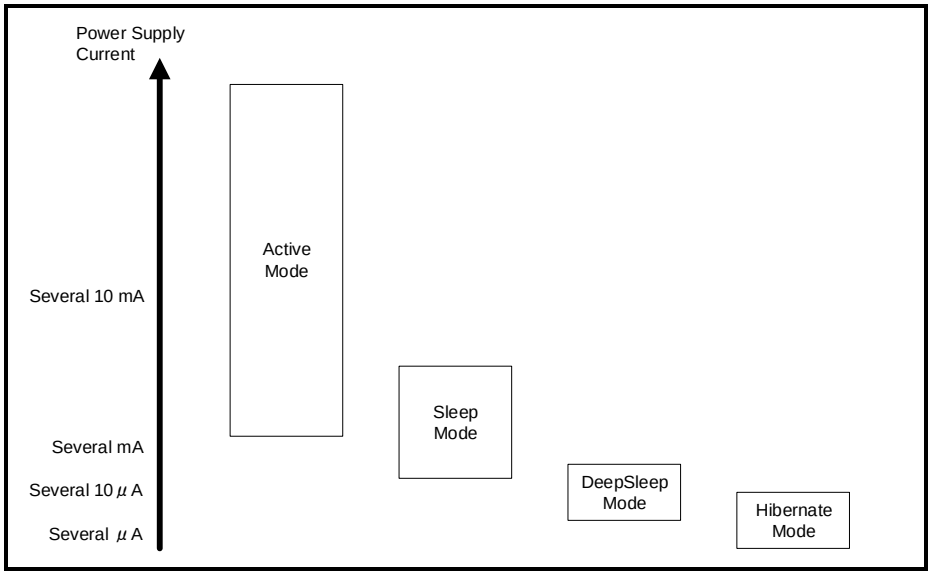
2 Power Modes of Traveo II Family

Traveo II family MCUs have the following power modes:

- Active mode: All peripherals are available.
- Sleep mode: All peripherals except the CPU are available.
- DeepSleep mode: Only low-frequency peripherals are available.
- Hibernate mode: Device and I/O states are frozen.

Figure 1 shows the relationship between power modes and the power supply current.

Figure 1. Power Modes and Power Supply Current



Note: Figure 1 is only an indication of the degree of power supply currents for each mode. Actual current values depend on the clock configuration and peripheral setting in each mode. For more details on power supply current characteristics, see the Datasheet.

Power consumption is reduced in the order of Active, Sleep, DeepSleep, and Hibernate. Each power mode optimizes power consumption for user applications.

Table 1 summarizes the states of each power mode and the entry and wakeup conditions. For more details on power modes, see Architecture TRM.

Table 1. Traveo II Power Modes

Power Mode	Description	Entry Condition	Wakeup Source	Wakeup Action
Active	Primary mode of operation; all peripherals are available (programmable).	Wake up from Sleep/DeepSleep modes, Hibernate reset, or any other reset.	Not applicable	Not applicable
Sleep	CPU is in Sleep mode; all other peripherals are available.	Register write from Active mode or wake up from DeepSleep through debugger	Any interrupt to CPU	Interrupt
DeepSleep	All high-frequency clocks and peripherals are turned OFF. Low-frequency clock (32 kHz) and low-power analog and digital peripherals are available for operation and as wakeup sources. SRAM can be retained (configurable).	Register write from Active modes or Debugger session ends.	GPIO interrupt, Event generators, SCB, watchdog timer, and RTC alarms [1] and debugger	Interrupt or debug
Hibernate	GPIO states are frozen. Almost all peripherals and clocks in the device are turned OFF. Device resets on wakeup.	Register write from Active mode.	WAKEUP pin and RTC alarms	Hibernate Reset

[1] RTC (along with WCO) is supplied with VDDD and is available irrespective of the device power mode. RTC alarms can wake up the device from any power mode.

Traveo II family MCUs have the following features:

- Software can use power modes to optimize power consumption in an application
- Low-power DeepSleep mode with support for multiple wakeup sources and configurable amount of SRAM retention
- Ultra-low-power Hibernate mode with wakeup from I/O and RTC alarms

The power consumption in different power modes is controlled by using the following methods:

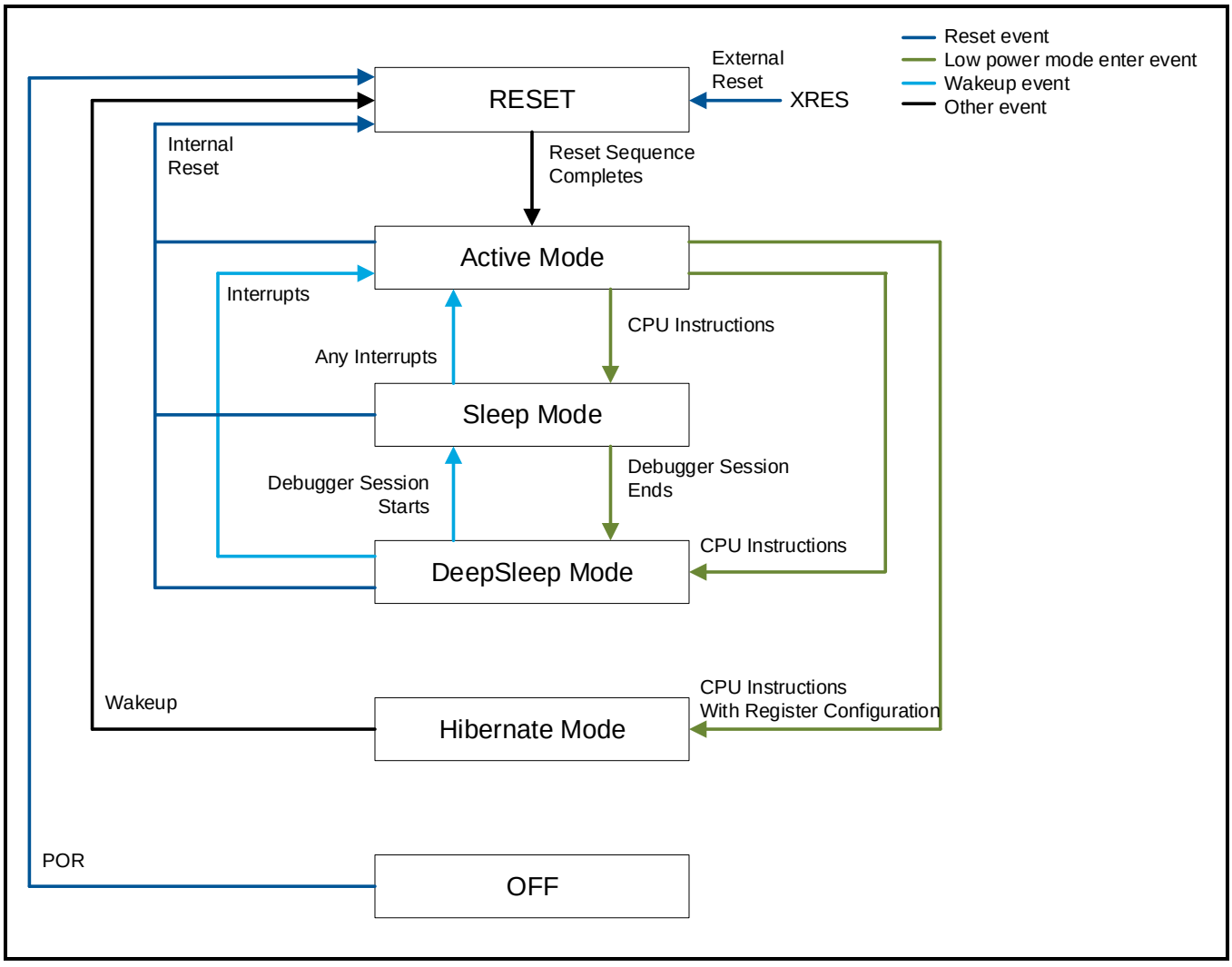
- Enabling and disabling clocks to peripherals
- Powering ON/OFF clock sources
- Powering ON/OFF peripherals and parts inside the MCUs

3 Power Modes Transition

3.1 Entering Power Modes

Figure 2 shows various states the device can be in along with possible power mode transition paths. The transitions are described in detail later in this application note.

Figure 2. Power Mode Transitions



3.1.1 RESET/OFF State

- OFF state:
  - Represents the state with no power applied
  - Go to RESET, when powered up above Power-On reset level (POR event)
- RESET state:
  - Detected reset event: POR, External Reset (XRES), or Internal Reset
  - Go to Active Mode after reset sequence completion
  - IMO is started
  - Device will enter RESET state upon assertion of XRES in any of the power modes

3.1.2 Entering Low Power Mode

Table 2 shows how to enter low-power mode, and the actions in low power modes.

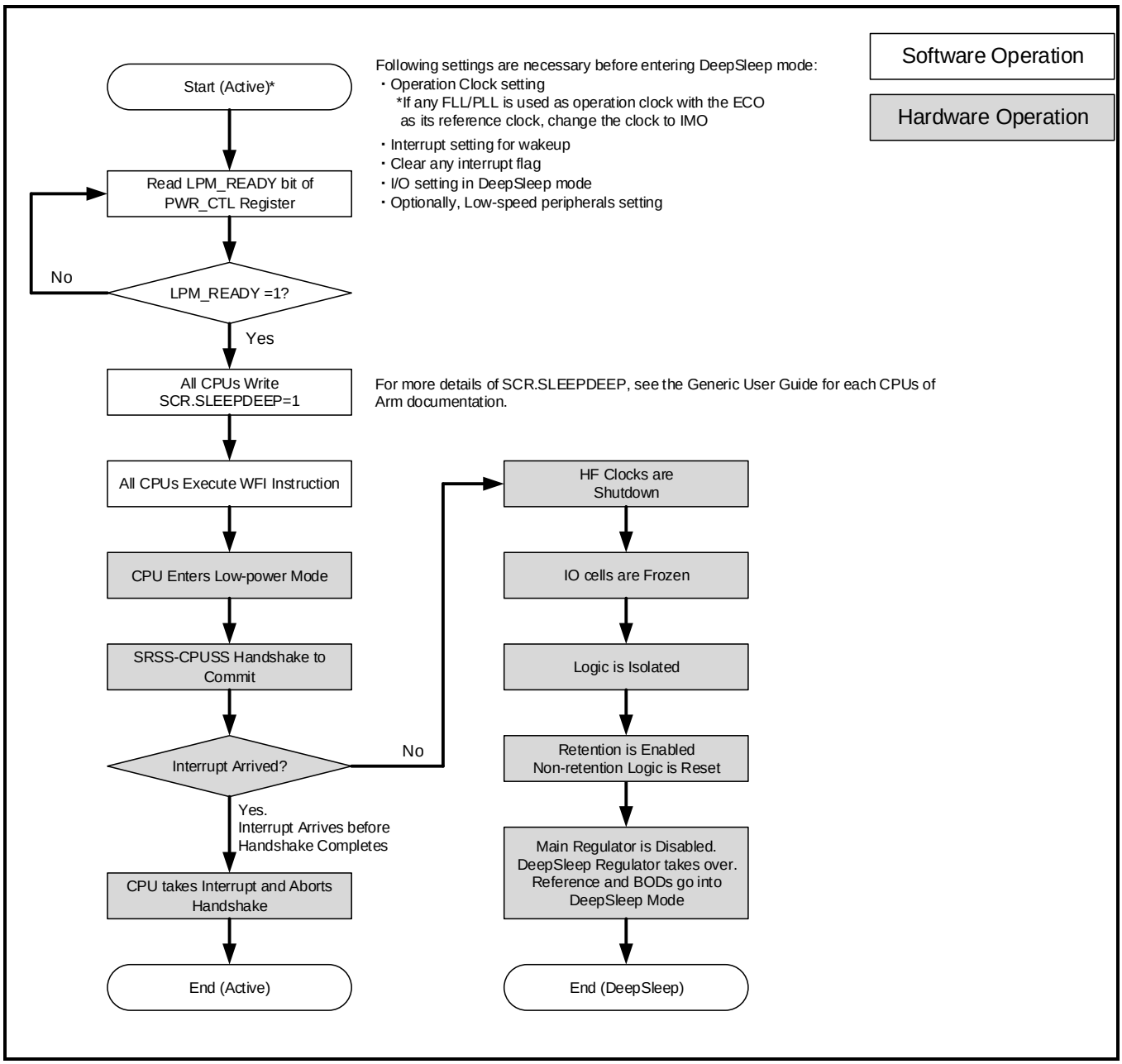
Table 2. Low-Power Mode Transitions

Initial State	Final State	Trigger	Hardware Actions
Active	Sleep	**Firmware action** 1. Clear the SLEEPDEEP bit [2] of the SCR register for all CPUs (For CYT2, CPUs are CM4 and CM0+. For CYT3/CYT4, CPUs are CM7 and CM0+). 2. Optionally, set the SLEEPONEXIT bit [1] of the SCR register, if the CPU runs only on interrupts. When this bit is set, the CPU will not return to application code after the WFI/WFE instruction is executed. The CPU will wake up on any enabled interrupt or event and will enter Sleep/DeepSleep mode as soon as it exits the interrupt or services the event. 3. Optionally, set the SEVONPEND bit [4] of the SCR register if the application needs to wake up the CPU from any pending interrupt. If this bit is set, any interrupt that enters a pending state will wake up the CPU. 4. Execute WFI/WFE instruction on all of CPUs. (For CYT2, CPUs are CM4 and CM0+. For CYT3/CYT4, CPUs are CM7 and CM0+).	1. CPU clocks are gated OFF. 2. CPU waits for an interrupt or event to wake it up.
Active	Deep Sleep	**Firmware action** Perform these steps to enter DeepSleep mode (LPM_READY bit [5] of the PWR_CTL register should read '1' before performing these steps): 1. Set the SLEEPDEEP bit [2] of the SCR register for all CPUs (For CYT2, CPUs are CM4 and CM0+. For CYT3/CYT4, CPUs are CM7 and CM0+). 2. Optionally, set the SLEEPONEXIT bit [1] of the SCR register, if the CPU runs only on interrupts. When this bit is set, the CPU will not return to application code after the WFI/WFE instruction is executed. The CPU will wake up on any enabled interrupt or event and will enter Sleep/DeepSleep mode as soon as it exits the interrupt or services the event. 3. Optionally, set the SEVONPEND bit [4] of the SCR register if the application needs to wake up the CPU from any pending interrupt. If this bit is set, any interrupt that enters a pending state will wake up the CPU. 4. Execute WFI/WFE instruction on all of CPUs (For CYT2, CPUs are CM4 and CM0+. For CYT3/CYT4, CPUs are CM7 and CM0+). **Note:** Executing the above sequence before the low-power mode is ready (LPM_READY==1) will transition first to Sleep mode. The device state will automatically move to DeepSleep state once LPM_READY bit is set. **Note:** Make sure that any write transfer made before executing the WFI instruction is followed by the read access to the same memory location. This ensures that the write operation is successful.	1. CPU enters low-power mode. 2. High-frequency clocks are shut down. 3. I/O cells will be frozen automatically. 4. Retention is enabled and non-retention logic is reset. 5. Active regulator is disabled and DeepSleep regulator takes over.

Initial State	Final State	Trigger	Hardware Actions
Active	Hibernate	**Firmware action** 1. Set TOKEN bits [7:0] of the PWR_HIBERNATE register (optional) and PWR_HIB_DATA register to some application-specific branching data that can be used on a wakeup event from Hibernate mode. 2. Set UNLOCK bits [8:15] of the PWR_HIBERNATE register to 0x3A for FREEZE and HIBERNATE bits of the PWR_HIBERNATE register to operate. 3. Configure wakeup pins polarity (POLARITY_HIBPIN bits [23:20]), wakeup pins mask (MASK_HIBPIN bits [27:24]) and wakeup alarm mask (MASK_HIBALARM bit [18]) in the PWR_HIBERNATE register based on the application requirement. 4. Set FREEZE bit [17] of the PWR_HIBERNATE register to freeze the I/O pins. 5. Set HIBERNATE bit [31] of the PWR_HIBERNATE register to enter Hibernate mode. 6. Read the PWR_HIBERNATE register to make sure that the write has taken effect. 7. Execute WFI instruction on all of CPUs. **Note**: It is recommended to trigger Hibernate mode atomically. That means, when entering Hibernate mode, disable all the interrupts and do a write operation on the PWR_Hibernate register. **Note:** Make sure that any write transfer made before executing the WFI instruction is followed by the read access to the same memory location. This ensures that the write operation is successful.	1. CPU enters low-power mode. 2. Both high-frequency and low-frequency clocks are shut down. 3. Retention is enabled and non-retention logic is reset. 4. Both Active and DeepSleep regulators are powered down. The peripherals that are active in the hibernate domain operate directly out of VDDD. 5. I/O cells are frozen
Sleep	Deep Sleep	When the debugger is not connected and DeepSleep mode is triggered, but LPM_READY==0, the device internally enters Sleep mode. The device will automatically transit to DeepSleep when LPM_READY==1. If the debugger is connected and DeepSleep mode is triggered by the firmware, the device will enter DeepSleep only when the following conditions are met: 1. LPM_READY==1 2. Debugger is disconnected	1. High-frequency clocks are shut down. 2. I/O cells will be frozen automatically. 3. Retention is enabled and non-retention logic is reset. 4. Active regulator is disabled and DeepSleep regulator takes over.

Figure 3 shows the software and hardware operation for the transition from Active mode to DeepSleep mode.

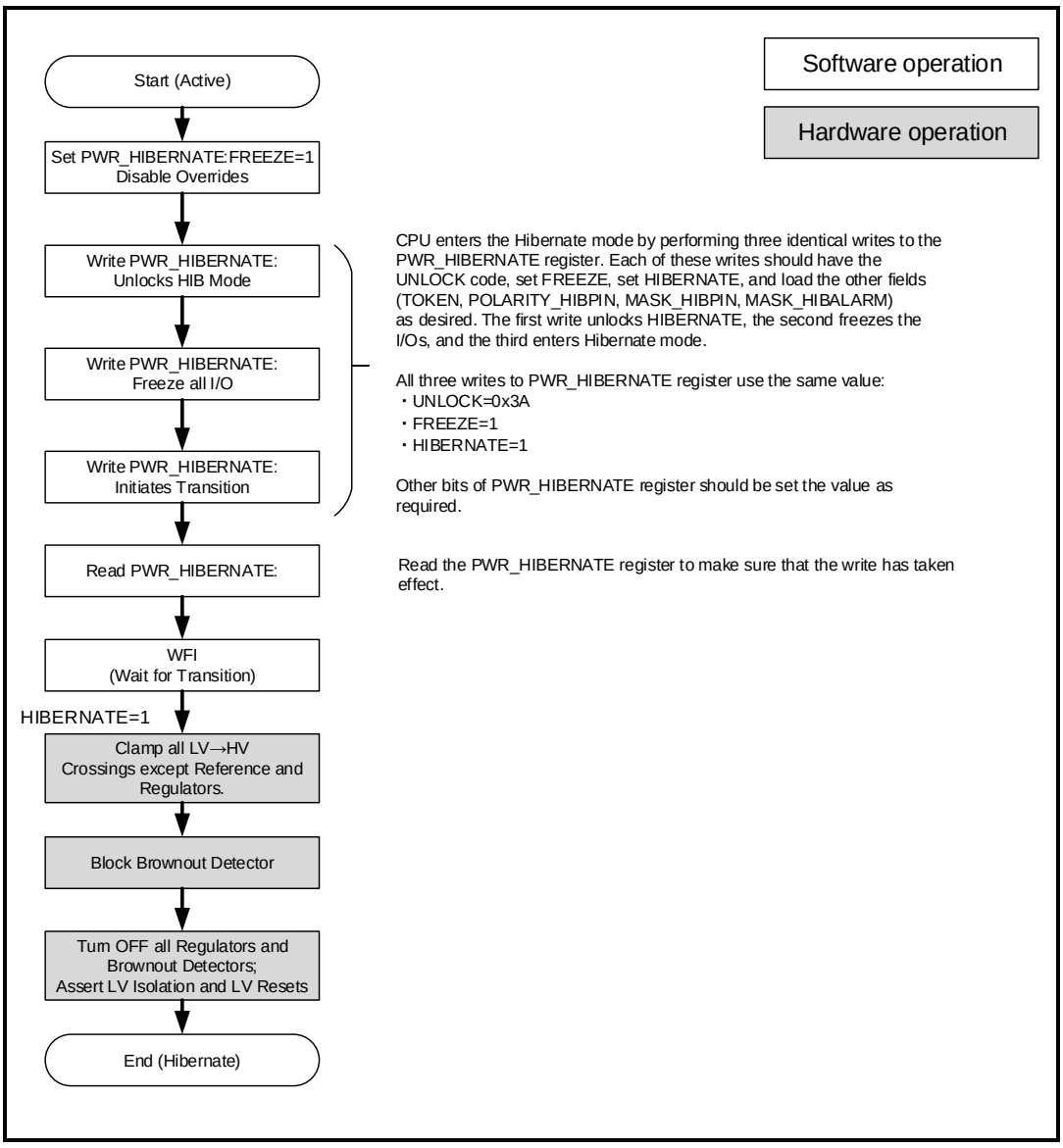
Figure 3. Active Mode to DeepSleep Mode Transition



Note: In Figure 3, the gray boxes indicate hardware operation. Therefore, processing with software is not required.

Figure 4 shows the software and hardware operation for the transition from Active mode to Hibernate mode.

Figure 4. Active Mode to Hibernate Mode Transition



Note: The gray boxes indicate hardware operation in Figure 4. Therefore, processing with software is not required.

3.1.3 Wakeup from Low-Power Modes

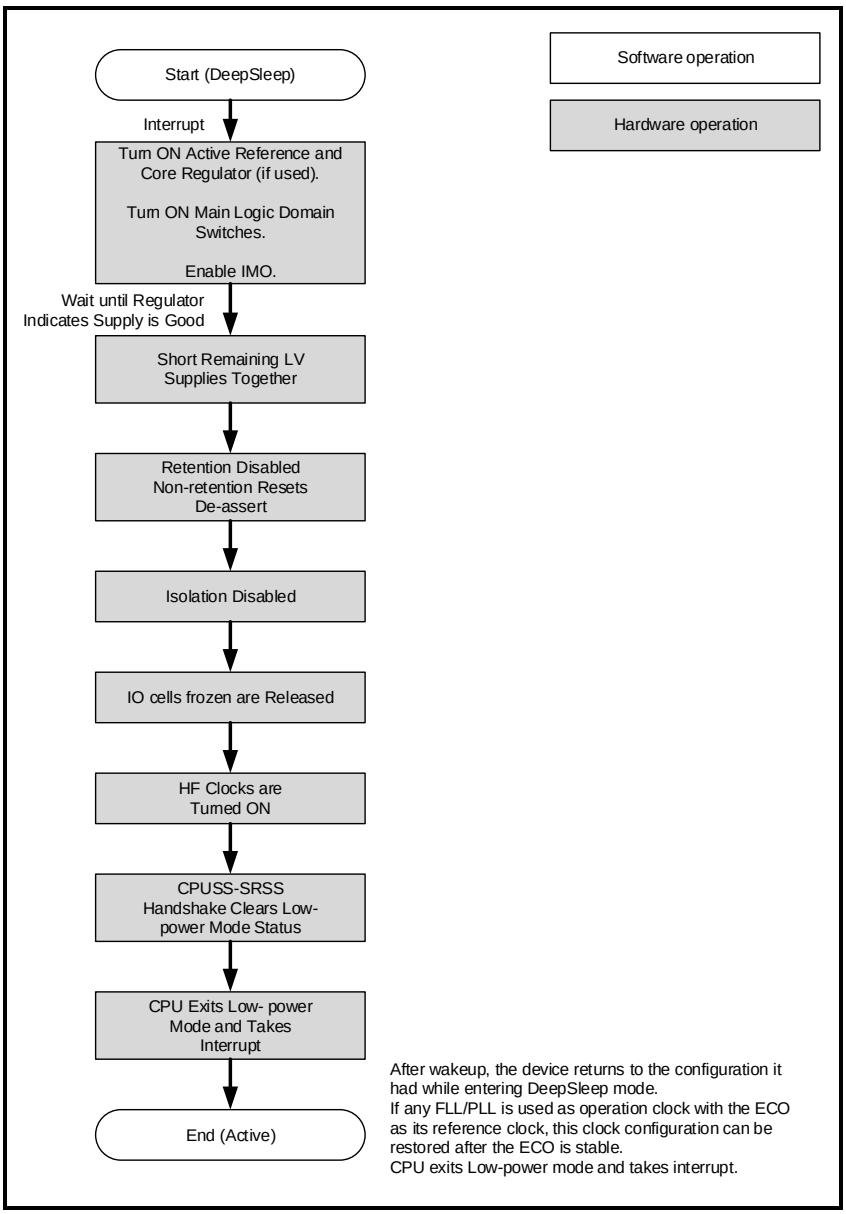
Table 3 shows the hardware triggers for wakeup and the actions after wakeup.

Table 3. Wakeup Action

Initial State	Final State	Trigger Source	Hardware Action
Sleep	Active	Any enabled interrupt in Sleep mode	CPU exits Sleep mode and executes the interrupt
DeepSleep	Active	Any enabled Interrupt in DeepSleep mode	Device returns to the configuration it had while entering DeepSleep mode. (IMO/clocks enabled, retention disabled, non-retained resets, freeze release; CPU exits low-power mode and takes interrupt)
DeepSleep	Sleep	Debug wakeup	Retention disabled and non-retained reset Freeze release HF and LF are ON CPU remains in Sleep state
Hibernate	Active	Wakeup pins, RTC alarms	Hibernate wakeup is implemented as a transition to Active mode through reset: 1. Low-voltage (internal Active and DeepSleep mode) regulators and references are ramped up 2. All low-voltage logic (operating from internal regulators) is reset 3. IMO clock is started 4. Core starts execution

Figure 5 shows the software and hardware operation for the transition from DeepSleep mode to Active mode.

Figure 5. DeepSleep Mode to Active Mode Transition



Note: The gray boxes indicate hardware operation in Figure 5. Therefore, processing with software is not required.

3.2 WDT Setting During Low-Power Modes

The watchdog timer (WDT) in Traveo II automatically resets the device in the event of an unexpected software execution path. In addition, the WDT can be used as an interrupt source or a wakeup source in Low-power modes. Software can select the resets or interrupts.

This section describes WDT setting and operation in low-power modes. For more details on the WDT, see the Architecture TRM.

3.2.1 Features

Traveo II supports two types of WDT: Basic WDT and Multi-counter WDT (MCWDT). Table 4 shows the supported WDT settings during Low-power modes.

Table 4. Supported WDT Settings During Low-Power Modes

Power Mode	Basic WDT	MCWDT			Remarks
		Subcounter0	Subcounter1	Subcounter2	
Active	Reset [2] and Interrupt [3]	Reset [2], Interrupt [3], and FAULT [4]		Interrupt [5]	In Active mode, the WDT can send the interrupt to the CPU.
Sleep	Reset [2] and Interrupt [3]	Reset [2], Interrupt [3], and FAULT [4]		Interrupt [5]	In Sleep mode, the CPU subsystem is powered down. Therefore, the interrupt request from the WDT is directly sent to the wakeup interrupt controller (WIC), which will then wake up the CPU.
DeepSleep	Reset [2] and Interrupt [3]	Reset [2], Interrupt [3], and FAULT [4]		Interrupt [5]	In DeepSleep mode, the CPU subsystem is powered down. Therefore, the interrupt request from the WDT is directly sent to the WIC, which will then wake up the CPU. Pauses/runs the counter is selectable during DeepSleep mode.
Hibernate	Reset [2] and Interrupt [3]	Not Supported			■ Can pause or run the counter; this option is selectable during Hibernate mode. ■ In Hibernate mode, any interrupt to wake up the device results in reset.

3.2.2 Example of WDT Wakeup Operation

Figure 6 shows an example of the operation with Subcounter0/1 of MCWDT. In this example, Subcounter0 of MCWDT is used as a supervisor of an unexpected software execution path, and Subcounter1 of MCWDT is used as a periodic wakeup interrupt generator during Low-power mode. For more details on the WDT, see Architecture TRM.

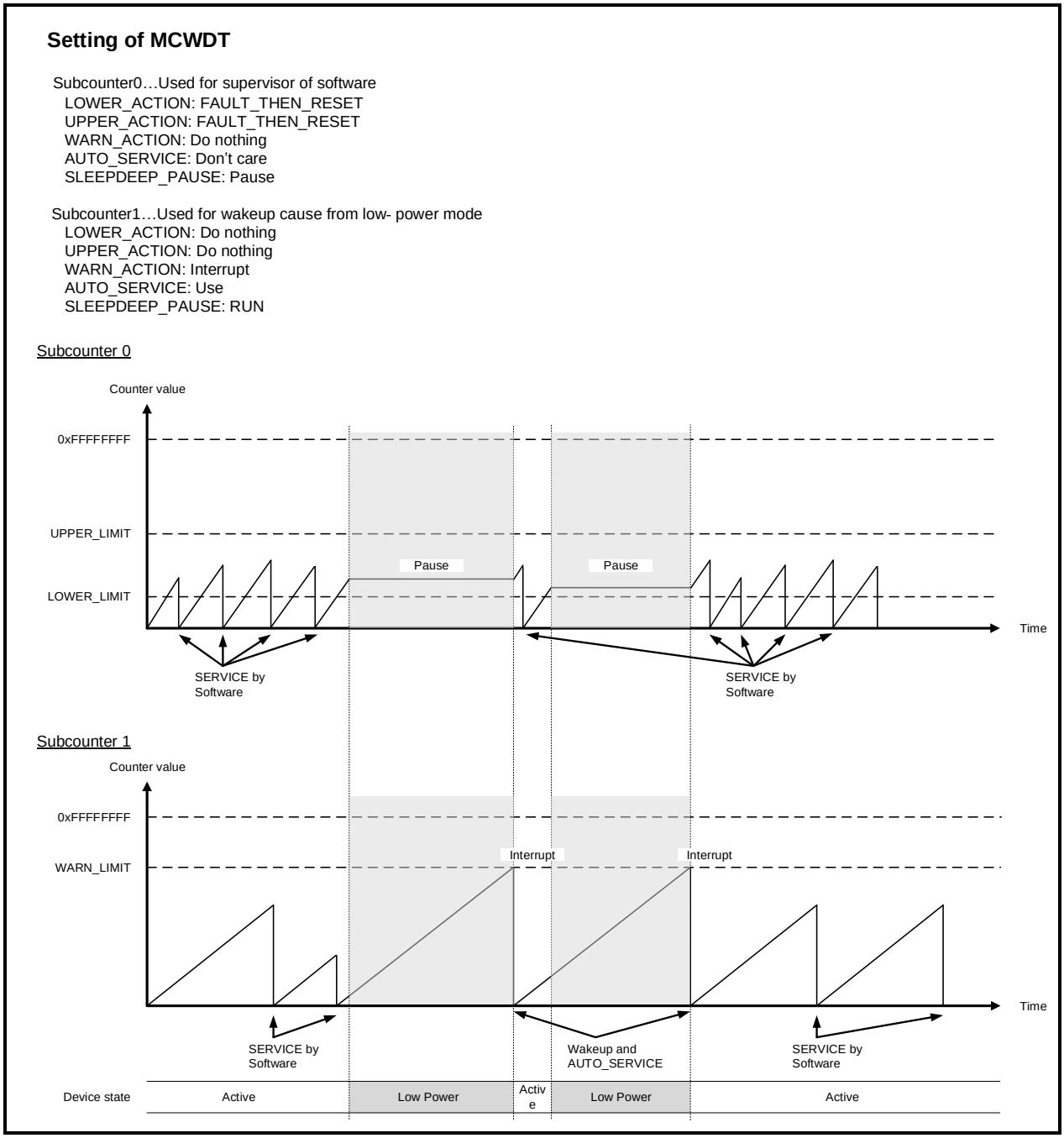
[2] Reset occurs when the counter value reaches UPPER_LIMIT or when the counter is cleared before LOWER_LIMIT.

[3] Interrupt occurs when the counter value reaches WARN_LIMIT.

[4] The fault manager converts this to a high-priority interrupt (such as Non-Maskable Interrupt, NMI) that gives the processor an opportunity to return to a safe state, such as halting memory writes and releasing peripherals.

[5] Interrupt occurs when the BIT specified by MCWDT2_CTR2_CONFIG [20:16] toggles.

Figure 6. Example of WDT Operation (Wakeup cause is interrupt of WARN_LIMIT of Subcounter1)



Subcounter0 is paused during Low-power mode. If MCU wakes up from Low-power mode, Subcounter0 resumes counting upwards.

Subcounter1 continues counting upwards during Low-power mode. If counter value reaches the setting value of "WARN_LIMIT", MCU wakes up from Low-power mode. If AUTO_SERVICE setting is used, hardware resets the counter value.

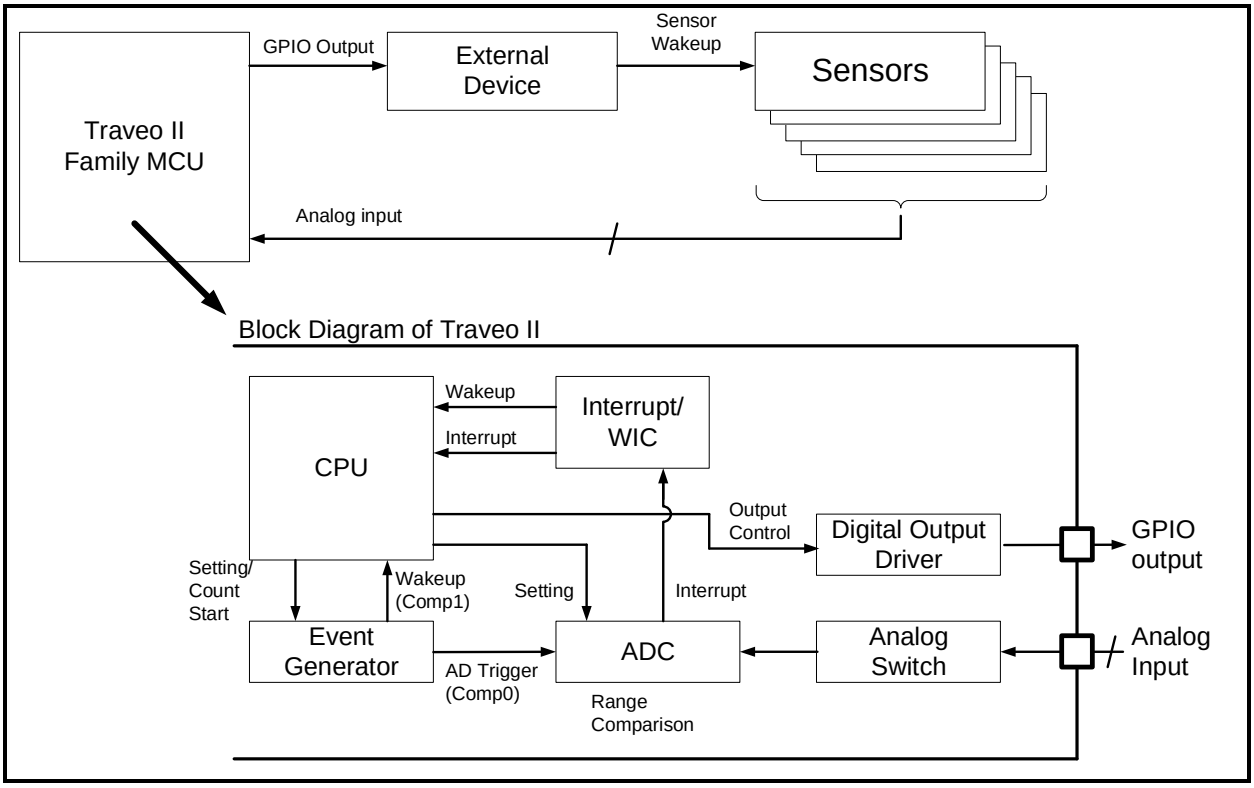
3.3 Cyclic Wakeup Operation

Cyclic Wakeup Operation is an intermittent MCU operation. For example, when the Electronic Control Unit (ECU) is in Sleep mode, MCU cyclically enters DeepSleep mode and wakes up. This operation is intended to minimize the average power consumption in an application. This section describes an implementation example of Cyclic Wakeup Operation by using Traveo II family MCUs.

3.3.1 Usage Example of Cyclic Wakeup

Figure 7 shows an example of a user system. MCU controls the GPIO and ADC for monitoring external devices, sensors, and so on.

Figure 7. Block Diagram of an Example User System



The external device is connected to MCU via GPIO. The external device wakes up the sensor by the control signal from the MCU. The sensor outputs are connected to the ADC of the MCU.

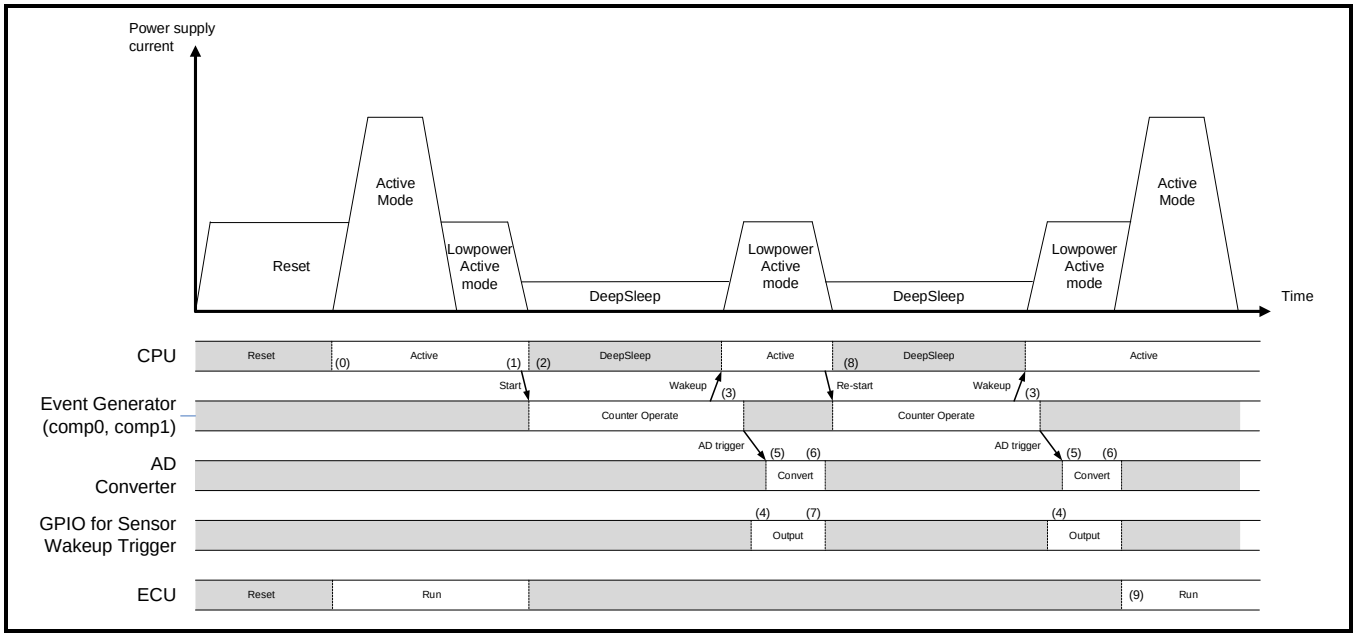
However, the sensor generally requires a stabilization wait time to correctly output after wakeup.

Therefore, the MCU outputs the sensor wakeup signal to the external device, and activates ADC after a certain time to convert the sensor signal. For generating these two different timings, two timer interrupts (Comp 0 and Comp 1) of the Event Generator are used. In this case, Comp 0 is used as a trigger for ADC activation, and Comp1 is used for CPU wakeup.

3.3.2 Cyclic Wakeup Operation

Figure 8 shows the concept of the Cyclic Wakeup Operation. Traveo II device cyclically wakes up to check the external sensor information when the ECU enters Sleep mode.

Figure 8. Cyclic Wakeup Operation



(0) After a reset, Traveo II enters Active mode and operates the user software.

(1) The CPU core configures and runs the Event Generator with the 32.768 kHz low-frequency oscillator.

(The source clock of the Event Generator can be selected from ILO0, ILO1, and WCO.)

(2) Traveo II enters DeepSleep mode; the CPU core goes to DeepSleep state.

(3) If the counter value of the Event Generator matches Comp1, the Comp1 trigger wakes up the CPU core.

(4) The CPU (software) controls the GPIO to output a wakeup trigger activation for external devices.

(5) Comp0 trigger starts an ADC range comparison.

(6) The CPU (software) observes ADC results.

(7) The CPU (software) controls the GPIO.

[If ADC results of range detection are in range, ECU continues to run Cyclic Wakeup Operation.]

(8) The CPU (software) restarts the Event Generator. The CPU goes back to DeepSleep mode. [Go to (2)]

[If ADC results of range detection are out of range, ECU exits from Cyclic Wakeup Operation.]

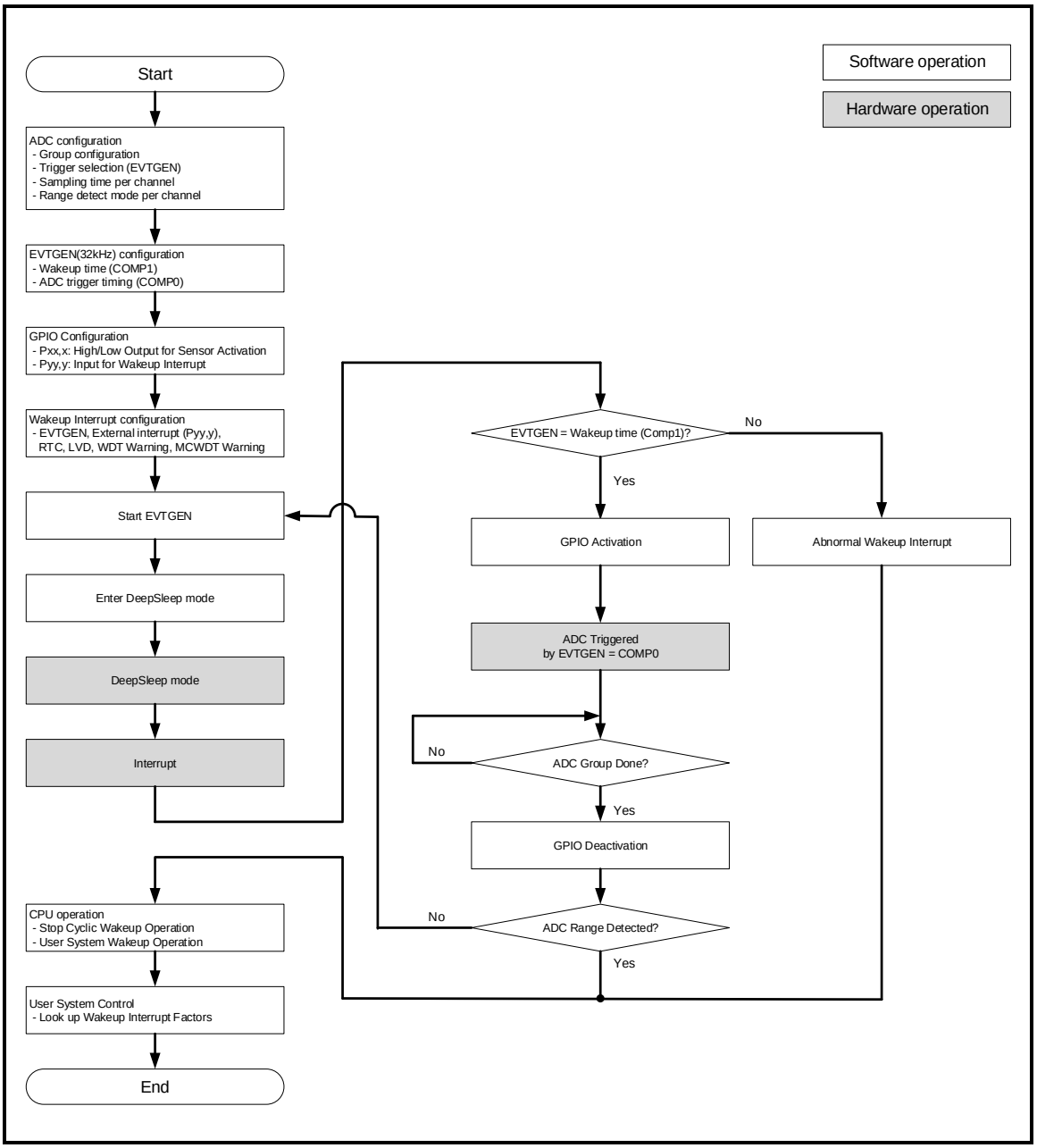
(9) CPU (software) restarts the operation of the ECU system.

For more details on Event generator, ADC, GPIO and Clock, see Architecture TRM.

3.3.3 Flowchart of Cyclic Wakeup Operation

Figure 9 shows the flowchart of the Cyclic Wakeup operation.

Figure 9. Flowchart of Cyclic Wakeup Operation



Note: The gray boxes indicate hardware operation in Figure 9. Therefore, processing with software is not required.

(1) Initial setting for using resources

- Initialized ADC, Event generator, GPIO, and wakeup interrupt configuration

(2) Event generator start

- Comp0 for ADC activation and Comp1 for CPU wakeup start to count.
- Comp0 is determined by the stabilization wait time for sensors.

- Comp1 is determined by the period time for cyclic wakeup.

(3) Enter DeepSleep mode

(4) Cyclic wakeup

- When wakeup trigger occurs, CPU will check whether interrupt is cyclic wakeup.
- If it is cyclic wakeup, CPU outputs control signal to external device to activate sensor.
- If it is not cyclic wakeup, CPU returns to Active mode as the abnormal interrupt occurrence.

(5) ADC activates by Comp0

- ADC converts the setting channels, and checks the conversion results with range comparator.
- If conversion result is within range, range comparator does not set flag.
- Otherwise, range comparator sets flag.

(6) Checking the sensor outputs

- CPU outputs control signal to external device to deactivate sensor after ADC conversion completion.
- CPU checks the flags of all conversion channels.
- If flags are set, CPU transfers to ACTIVE mode.
- If flags are not set, CPU transfers to DeepSleep mode again after restarting timers of Comp0 and Comp1.

3.3.4 Usage of Smart I/O in Cyclic Wakeup

This section describes the role of Smart I/O in reducing the LPACTIVE period in Cyclic Wakeup Operation.

As described in the beginning of the Cyclic Wakeup Operation section, Cyclic Wakeup is intended to minimize the average power consumption in an application. This average consumption current is affected by the following:

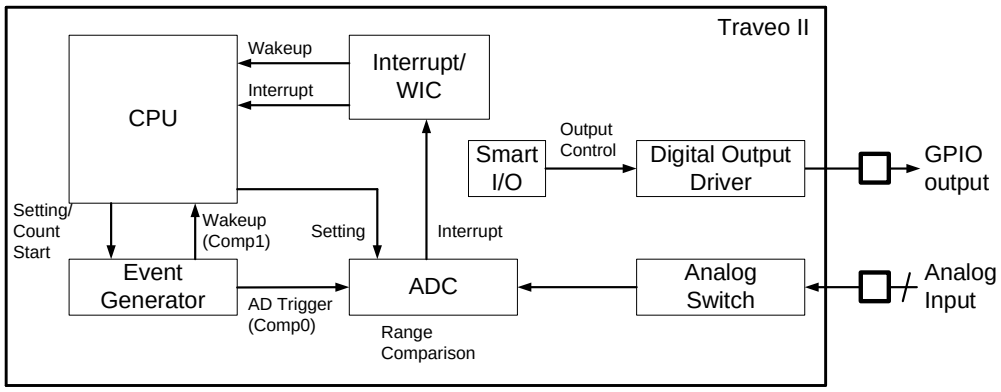
- DeepSleep current
- LPACTIVE current
- Percentage of LPACTIVE time in one period.

DeepSleep current and LPACTIVE current mostly depend on electrical specifications of HW, while the percentage of the LPACTIVE time in one period depends on the optimization of SW. Because the current consumption during LPACTIVE (several mA) is relatively much higher than in DeepSleep (several 10 μ A), from the SW point of view, reducing the LPACTIVE time plays an important factor in achieving a low average consumption current. To shorten the LPACTIVE period, use of Smart I/O is suggested.

In the flow of Cyclic Wakeup operation proposed in Figure 8, the MCU need to wake up and configure I/O ports to turn sensors on, and wait for the sensor's stabilization before triggering ADC conversion. If a long sensor stabilization time is required, the MCU can optionally be put in Sleep/DeepSleep again to reduce the current consumption, but this approach may make program more complex. Additionally, you should consider the transition time between different power modes.

Figure 10 shows the system configuration of this use case. GPIO is activated by Smart I/O, instead of CPU as demonstrated in Figure 7.

Figure 10. System Configuration of Cyclic Wakeup with Smart I/O



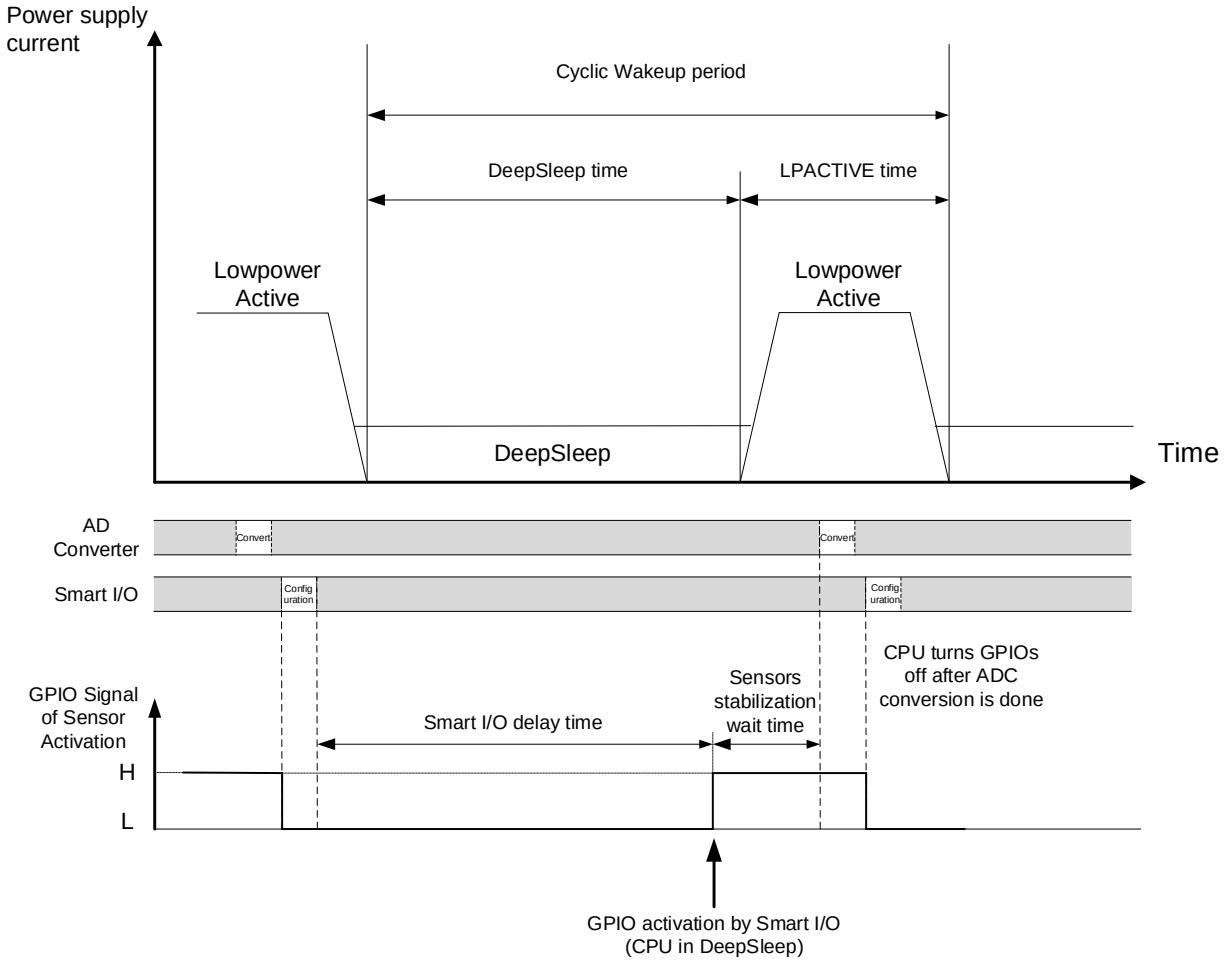
3.3.4.1 Advantage of Smart I/O Implementation in Cyclic Wakeup

Smart I/O can be used to manipulate I/O during DeepSleep mode. This can eliminate the unnecessary LPACTIVE time CPU spends to activate I/O especially when the CPU must wait for a long time for the sensor to stabilize.

The internal logic of Smart I/O includes three-input lookup table (LUTs) and Data Unit (DU) among other components. DU acts as a counter with count up/down and reload function. By setting Data Unit and LUTs properly, we can create a circuit that delays GPIO from outputting high with desired latency.

As shown in Figure 11, because Smart I/O can operate in DeepSleep, you can use it to activate GPIO while the CPU is in DeepSleep mode.

Figure 11. Smart I/O based Cyclic Wakeup Operation



3.3.4.2 Smart I/O Configuration in Cyclic Wakeup

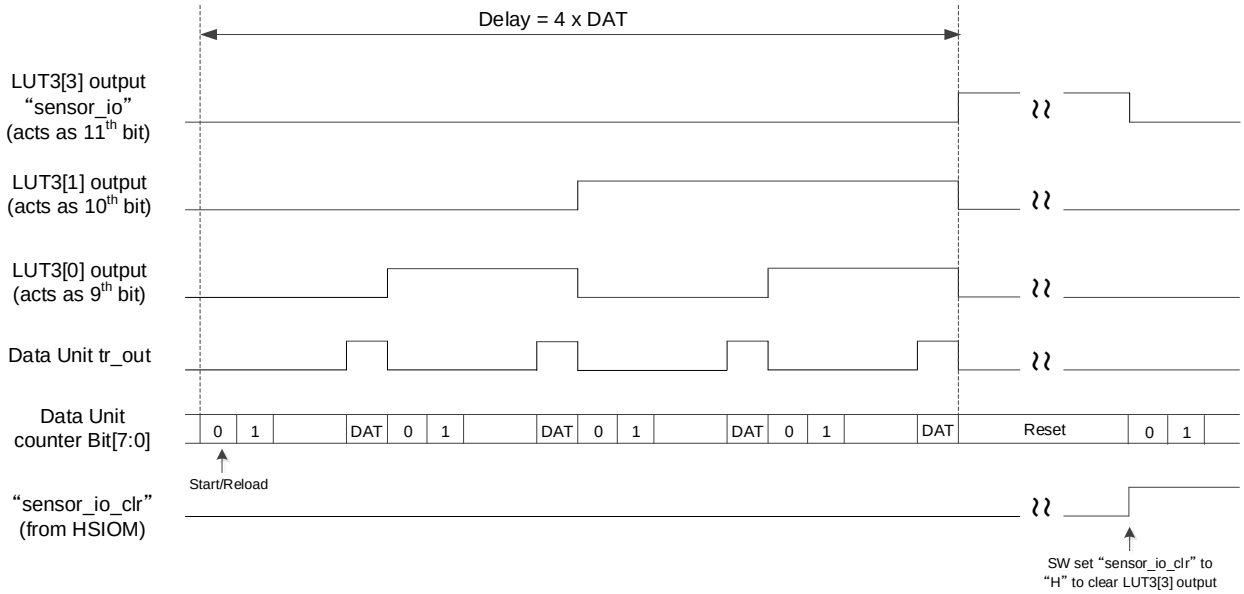
The Data Unit (DU) can be used as a counter to delay “H” output. However, the DU is only 8-bit counter. During DeepSleep, because the source clock of Smart IO is ILO with frequency of 32 kHz, the DU can count for maximum interval as follows:

$$\frac{2^8}{32 \times 10^3} \times 10^3 \; = \; 8\,[ms]$$

Therefore, for applications that require Cyclic Wakeup period larger than 8 ms, you need extra bits for the counter.

Thus, for this example, an 11-bit timer equivalent circuitry by Smart I/O called “Sensor Activation Circuitry” is implemented. An 11-bit counter equivalent circuitry can generate delay up to 32 ms. Figure 12 shows the operation of the circuitry. Here, ‘DAT’ is the upper limit of the DU. DAT is configured by the SMARTIO_PRTx_DATA register. A single clock pulse is output at the Data Unit tr_out when the count value is equal to DAT.

Figure 12. Operation of Sensor Activation Circuitry

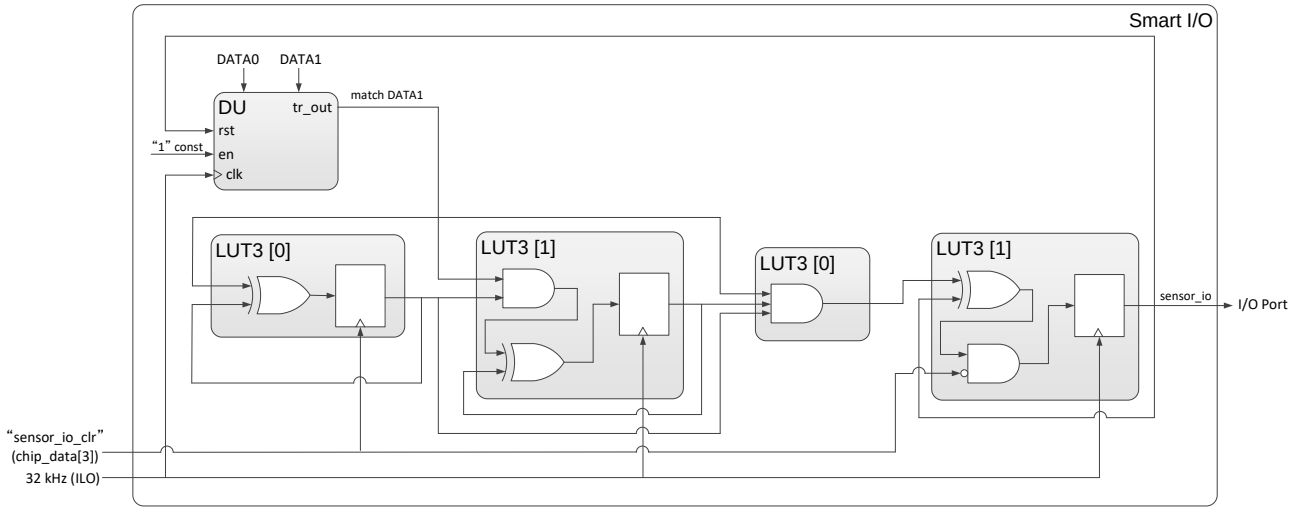


In this use case, the circuitry receives one signal from HSIOM named the "sensor_io_clr" signal, with active HIGH, which is used to clear the output of LUT3[3], i.e., the sensor activation output. The following I/O port and HSIOM signal are used:

- smartio_data[3] = sensor_io (to I/O port, i.e., external sensor activation port)
- chip_data[3] = sensor_io_clr (from HSIOM, manipulated by the CPU to clear 'sensor_io')

Figure 13 shows the connection and functional logic of each LUT3 and DU in this circuitry.

Figure 13. Logical Example of a Sensor Activation Circuitry



The Data Unit operates in INCR_WRAP mode. This mode increments the data by 1 from an initial value (DATA 0) until it reaches a final value (DATA 1). When the count value matches the final value, it wraps around to DATA 0.

In this circuitry, the Data Unit carries the lower 8 bits of the 11-bit counter, LUT3[0], LUT3[1], LUT3[3] stands for the 9th, 10th, and 11th bit of the 11-bit counter. The output of LUT3[3], i.e., the 11th bit of the counter, is connected to the GPIO port to activate the external sensor. Figure 14 shows the signal path in this use case.

Figure 14. Signal Path of Sensor Activation Circuitry

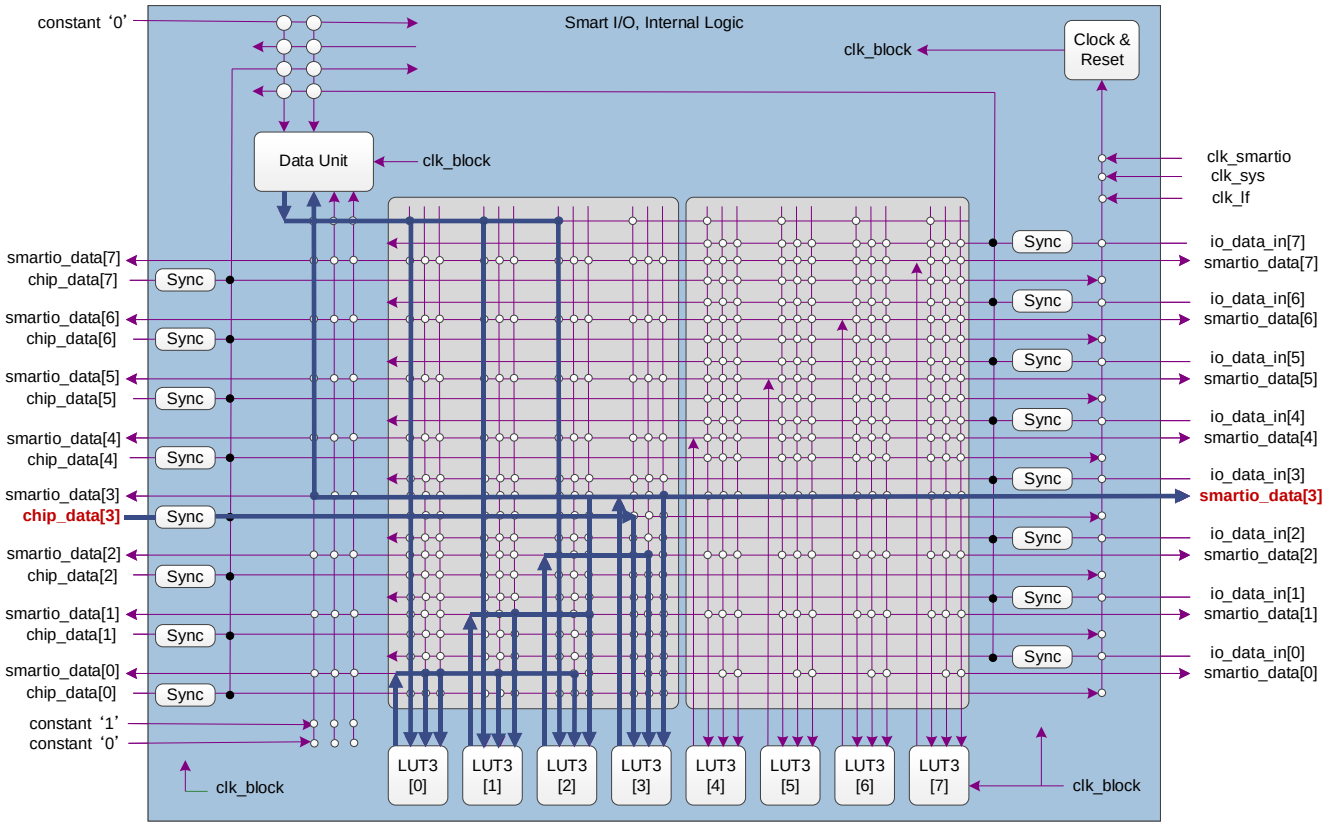


Table 5, Table 6, Table 7 and 8 show the truth table of each LUT3. The red highlights in the table indicate an invalid pattern.

Table 5. Look Up Table LUT3 [0]

Tr0_in	Tr1_in	Tr2_in	Tr_out
LUT3[0] out	LUT3[0] out	DU tr_out	
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Table 6. Look Up Table LUT3 [1]

Tr0_in	Tr1_in	Tr2_in	Tr_out
LUT3[1] out	LUT3[0] out	DU tr_out	
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Table 7. Look Up Table LUT3 [2]

Tr0_in	Tr1_in	Tr2_in	Tr_out
LUT3[1] out	LUT3[0] out	DU tr_out	
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Table 8. Look Up Table LUT3 [3]

Tr0_in	Tr1_in	Tr2_in	Tr_out
LUT3[3] out	LUT3[2] out	CHIP_ DATA[0]	
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

The H output delayed by the Sensor Activation Circuitry can be calculated as follows:

$$\text{delay} = 4 \times \frac{DAT1 - DAT0}{32} = \frac{DAT1 - DAT0}{8}[ms]$$

Therefore, you can configure DAT1 and DAT0 (usually set to '0') to satisfy the sensor stabilization waiting time as in the rough estimation as follows:

$$\text{delay} = T_{Cyclic\ Wakeup\ period} - T_{sensor\ stabilization\ wait}$$

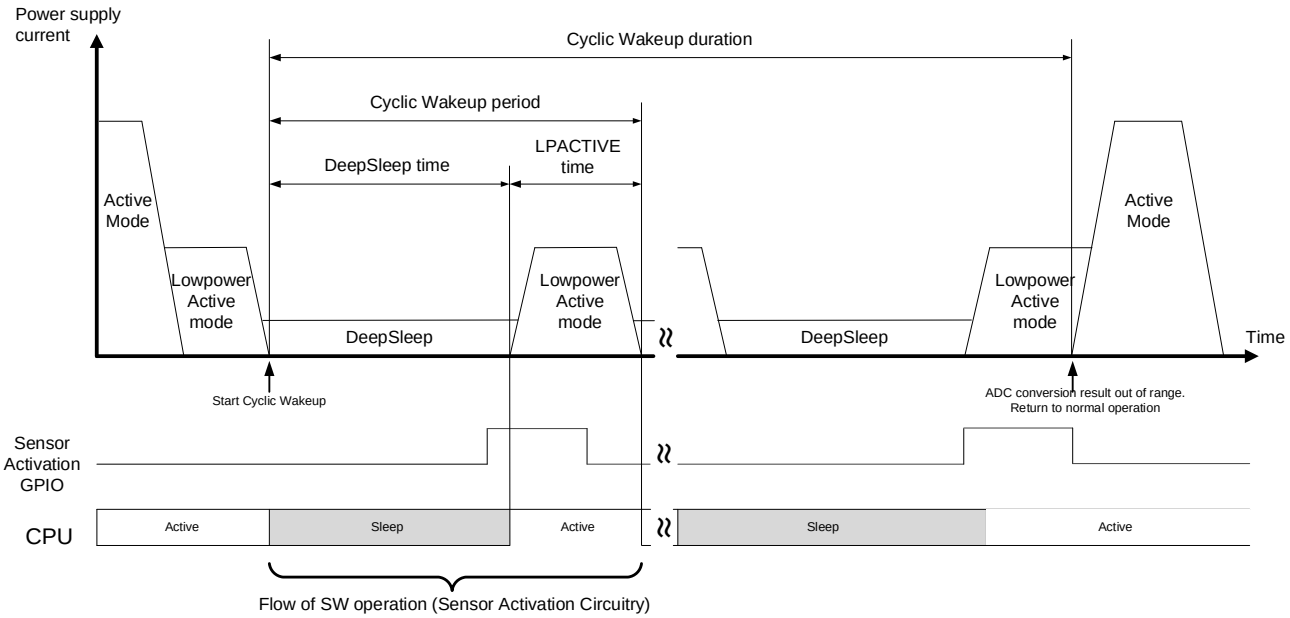
$$\therefore \text{DAT1} = 8 \times (T_{Cyclic\ Wakeup\ period} - T_{sensor\ stabilization\ wait})$$

For example, if $T_{Cyclic\ Wakeup\ period}$ =32 [ms], you can configure DAT0 = 0, DAT1 = 0xFD to make $T_{sensor\ stabilization\ wait} \geq 300[\mu s]$.

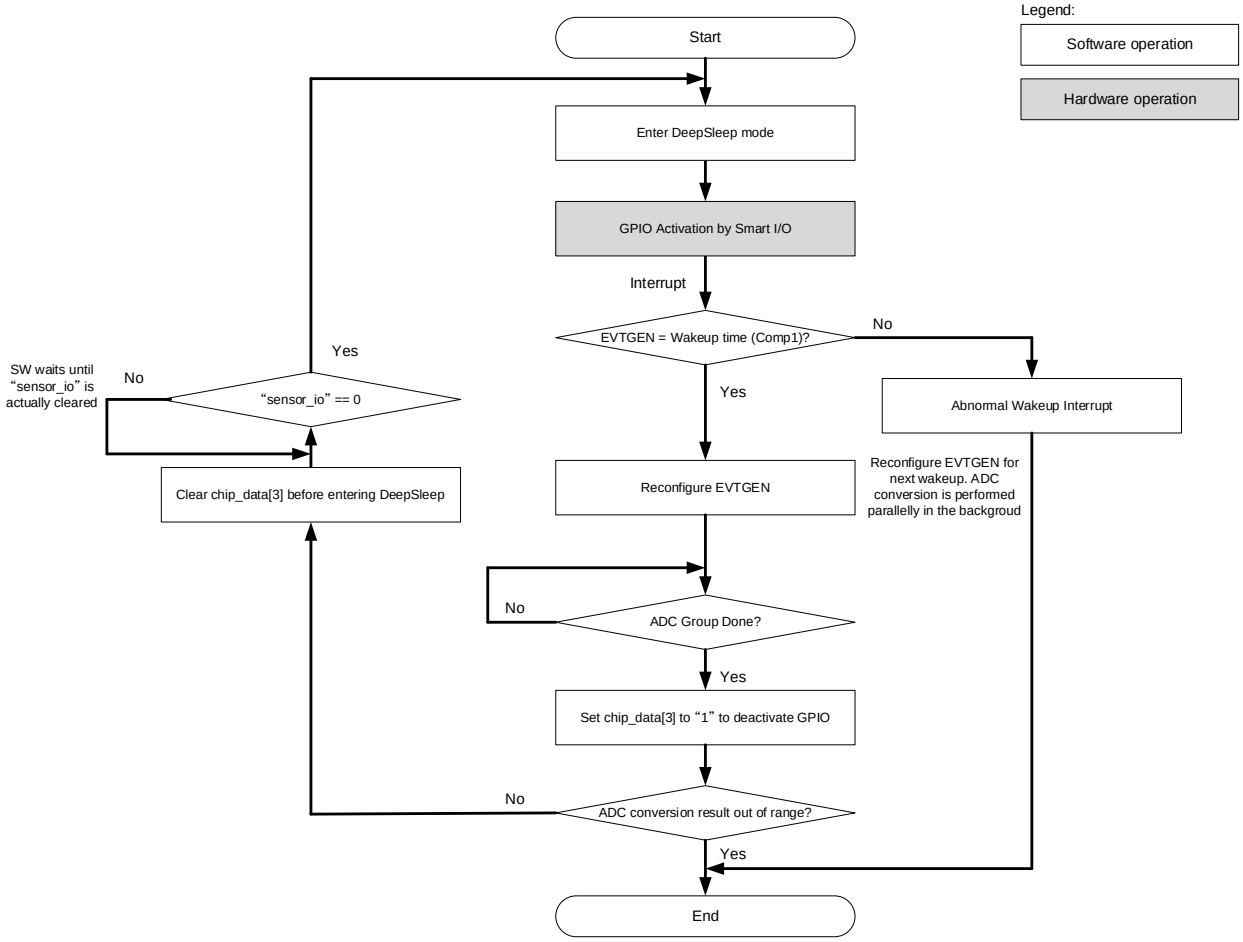
3.3.4.3 Sensor Activation Circuitry in Cyclic Wakeup Operation

Using the Sensor Activation Circuitry constructed in the previous section, the operation of Cyclic Wakeup can be enhanced as shown in Figure 15.

Figure 15. Cyclic Wakeup Operation with Smart I/O Usage



Sensor Activation Circuitry in Cyclic Wakeup



Note: The gray box in the flowchart indicates a hardware operation. Therefore, processing with software is not required.

The GPIO is activated by Smart I/O while the CPU is still in DeepSleep mode and the sensor's stabilization time can be satisfied just by adjusting DAT0 and DAT1 properly. The behavior of the GPIO is now isolated from the operation of the CPU; this makes the software flow less complex.

For example, the CPU doesn't need to go to Sleep mode after activating the GPIO to cut back the current consumption if a long sensor waiting time is required.

3.4 CAN Wakeup Operation

As long as there is any communication on the CAN bus, the ECU is awake. If CAN communication occurs while the ECU is in low-power mode, ECU must wake up from low-power mode. This section describes an implementation example of the CAN Wakeup operation by using Traveo II family MCUs.

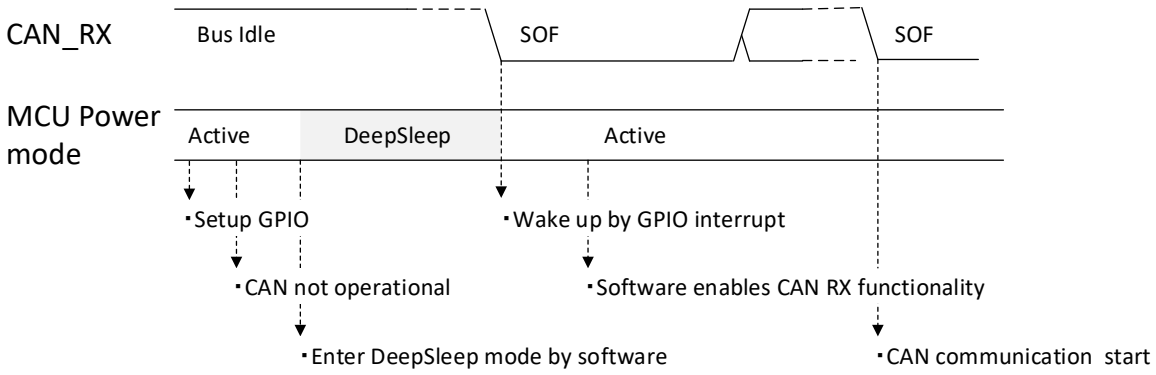
Traveo II family MCUs have the following pins for wakeup:

- Up to 4 pins to wake up from Hibernate mode
  See the datasheet for the supported number of pins that can wake up the device from Hibernate mode.
- GPIO pins to wake up from DeepSleep mode
  See the datasheet for the supported number of GPIO that can wake up the device from DeepSleep mode.

The CAN block cannot detect a wakeup condition when the MCU is DeepSleep or Hibernate mode. To support CAN Wakeup, MCU should use the function of GPIO interrupt or WAKEUP pin.

Figure 16 shows the example of CAN Wakeup from DeepSleep.

Figure 16. CAN Wakeup from DeepSleep Mode



The software sets the I/O port and GPIO interrupt before the MCU enters low power mode. The CAN receive functionality is disabled. After that, when the MCU detects a wakeup signal while in low-power mode, the MCU wakes up from low-power mode. After that, software enables the CAN receive functionality and CAN communication starts. For more details on CAN and GPIO, see Architecture TRM.

4 Glossary

Terms	Description
ADC	Analog-to-digital converter. See the “SAR ADC” chapter of the Architecture TRM for details.
Basic WDT	Basic Watchdog Timer. See the “Watchdog Timer” chapter of the Architecture TRM for details.
BOD	Brown-out detection. See the “Power Supply and Monitoring” chapter of the Architecture TRM for details.
CPUSS	CPU subsystem. See the “CPU Subsystem” section of the Architecture TRM for details.
ECO	External Crystal Oscillator. See the “Clocking System” chapter of the Architecture TRM for details.
EVTGEN	Event Generator. See the “Event Generator” chapter of the Architecture TRM for details.
FLL	Frequency Locked Loop. See the “Clocking System” chapter of the Architecture TRM for details.
GPIO	General-Purpose Input/Output. See the “I/O System” chapter of the Architecture TRM for details.
HF	High Frequency Clock. See the “Clocking System” chapter of the Architecture TRM for details.
ILO	Internal Low-speed Oscillators. See the “Clocking System” chapter of the Architecture TRM for details.
IMO	Internal Main Oscillator. See the “Clocking System” chapter of the Architecture TRM for details.
LF	Low Frequency Clock. See the “Clocking System” chapter of the Architecture TRM for details.
LPACTIVE	Low Power Active. See the “Device Power Modes” chapter of the Architecture TRM for details.
LV supplies	Low-Voltage supplies.
LVD	Low-Voltage Detection. See the “Power Supply and Monitoring” chapter of the Architecture TRM for details.
MCWDT	Multi-counter Watchdog Timer. See the “Watchdog Timer” chapter of the Architecture TRM for details.
Pending interrupt	Interrupt of pending state. See the “Interrupts” chapter of the Architecture TRM for details.
PLL	Phase Locked Loop. See the “Clocking System” chapter of the Architecture TRM for details.
POR	Power On Reset. See the “Rest System” chapter of the Architecture TRM for details.
RTC	Real-Time Clock. See the “Real-Time Clock” chapter of the Architecture TRM for details.
SCB	Serial Communications Block. See the “Serial Communications Block (SCB)” chapter of the Architecture TRM for details.
SRSS	System Resources Subsystem. See the “System Resources Subsystem” section of the Architecture TRM for details.
VDDD	Digital power supply.
WCO	Watch Crystal Oscillator. See the “Clocking System” chapter of the Architecture TRM for details.
WDT	Watchdog Timer reset. See the “Watchdog Timer” chapter of the Architecture TRM for details.
WFE	Wait For Event instruction
WFI	Wait For Interrupt instruction
WIC	Wakeup interrupt controller. See the “Interrupts” chapter of the Architecture TRM for details.
XRES	External reset I/O pin. See the “Reset System” chapter of the Architecture TRM for details.

5 Related Documents

The following are the Traveo II family series datasheets and Technical Reference Manuals. Contact Technical Support to obtain these documents.

- Device datasheet
  - CYT2B7 Datasheet 32-Bit Arm® Cortex®-M4F Microcontroller Traveo™ II Family
  - CYT2B9 Datasheet 32-Bit Arm® Cortex®-M4F Microcontroller Traveo™ II Family
  - CYT4BF Datasheet 32-Bit Arm® Cortex®-M7 Microcontroller Traveo™ II Family
  - CYT4DN Datasheet 32-Bit Arm® Cortex®-M7 Microcontroller Traveo™ II Family
  - CYT3BB/4BB Datasheet 32-Bit Arm® Cortex®-M7 Microcontroller Traveo™ II Family
- Body Controller Entry Family
  - Traveo™ II Automotive Body Controller Entry Family Architecture Technical Reference Manual (TRM)
  - Traveo™ II Automotive Body Controller Entry Registers Technical Reference Manual (TRM) for CYT2B7
  - Traveo™ II Automotive Body Controller Entry Registers Technical Reference Manual (TRM) for CYT2B9
- Body Controller High Family
  - Traveo™ II Automotive Body Controller High Family Architecture Technical Reference Manual (TRM)
  - Traveo™ II Automotive Body Controller High Registers Technical Reference Manual (TRM) for CYT4BF
  - Traveo™ II Automotive Body Controller High Registers Technical Reference Manual (TRM) for CYT3BB/4BB
- Cluster 2D Family
  - Traveo™ II Automotive Cluster 2D Family Architecture Technical Reference Manual (TRM)
  - Traveo™ II Automotive Cluster 2D Registers Technical Reference Manual (TRM)

Document History

Document Title: AN220222 - Low Power Mode Procedure in Traveo II Family

Document Number: 002-20222

Revision	ECN	Submission Date	Description of Change
**	6458043	02/28/2019	New Application Note.
*A	6724804	11/05/2019	Added descriptions of CAN wake up and target parts number (CYT4D series).
*B	6808279	03/12/2020	Changed target parts number (CYT2/ CYT4 series). Added target parts number (CYT3 series).
*C	7036591	12/03/2020	Added usage of Smart I/O in Cyclic Wakeup

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Code Examples | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
An Infineon Technologies Company
198 Champion Court
San Jose, CA 95134-1709