AN204452

# How to Set Up Watchdog Timer for Traveo™ Family

**Author: Kenichi Sunada**

**Associated Part Family: Traveo Family S6J3110/3120/3200/3310/3320/3330/3340/3350/3360/3370/3400 Series**

**Related Documents: For a complete list, see Related Documents.**

This application note describes how to set up the software and hardware watchdog timers for the Traveo™ family S6J3110/3120/3200/3310/3320/3330/3340/3350/3360/3370/3400 Series.

## 1 Introduction

This application note is intended for users of the Traveo family S6J3110/3120/3200/3310/3320/3330/3340/3350/3360/3370/3400 Series. It describes how to set up the software watchdog timer and hardware watchdog timer for these series.

## 2 Overview of Watchdog Timer

The watchdog timer is used to detect a runaway state caused by a user program. It includes software and hardware watchdog timers. Table 1 describes the main differences between these timers.

Table 1. Differences between Software Watchdog Timer and Hardware Watchdog Timer

| Item | Software Watchdog Timer | Hardware Watchdog Timer |
|---|---|---|
| Watchdog timer start | Register write by user program | Automatic start after releasing reset |
| Watchdog counter source clock selection | ▪ High-speed CR clock<br>▪ Low-speed CR clock<br>▪ Main clock | ▪ High-speed CR clock<br>▪ Low-speed CR clock |
| Register setting value write | User program | BootROM marker<br>(BootROM software is built-in firmware that is executed after reset and before execution of user program.) |
| Operation in PSS mode | Enable/disable control by user program | STOP |

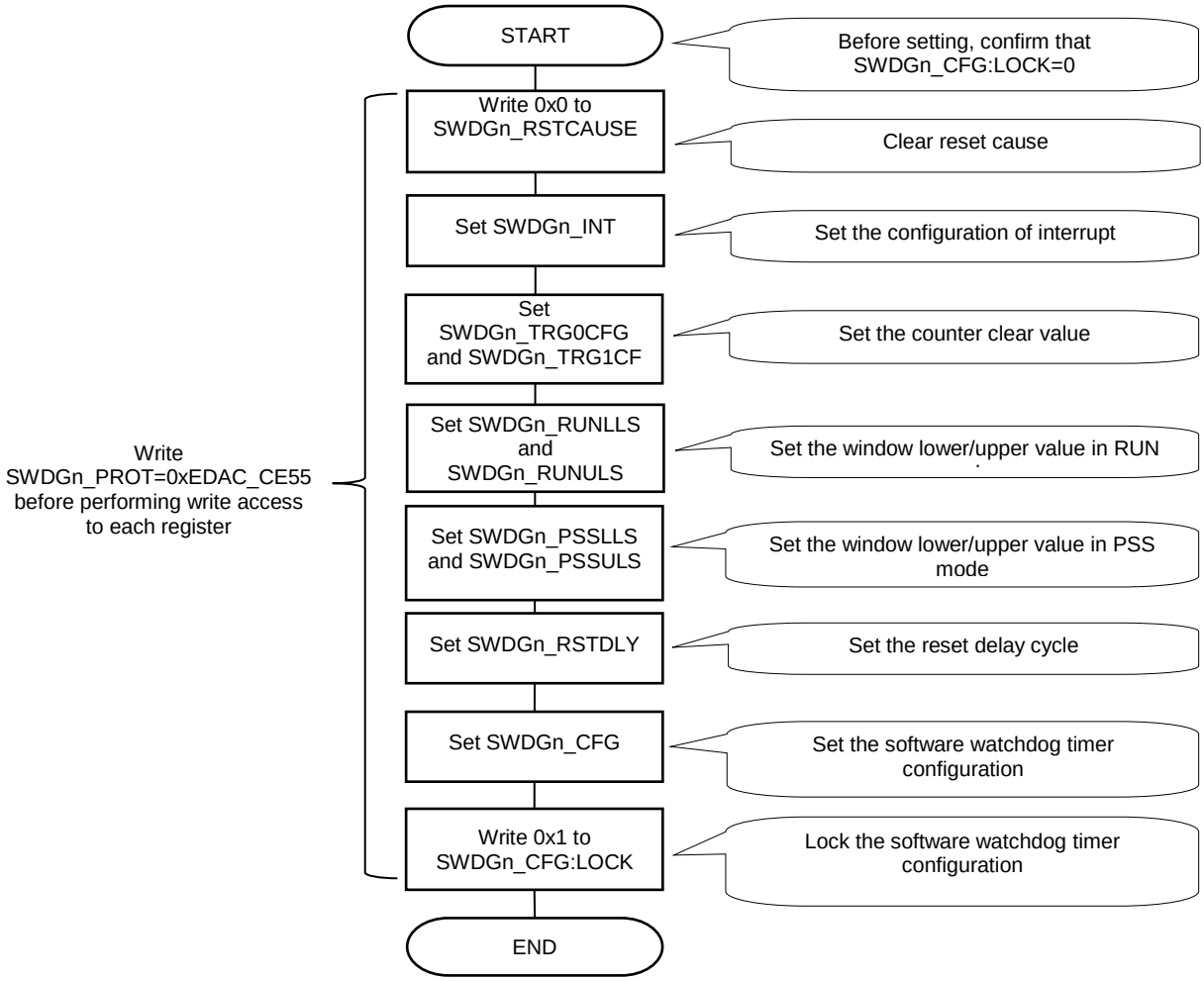
# 3 Watchdog Timer Setup Procedures

This section explains how to set up the software and hardware watchdog timers for the S S6J3110/3120/3200/3310/3320/3330/3340/3350/3360/3370/3400 Series.

## 3.1 Software Watchdog Timer Setup Procedure

Figure 1 depicts the flow of the example software watchdog timer setup procedure.

Figure 1. Example Software Watchdog Timer Setup Procedure Flow Chart



### 3.1.1 Software Watchdog Timer Setup Procedure of Sample Software

Figure 2 shows the software watchdog timer setup procedure of the sample software.

Figure 2. Software Watchdog Timer Setup Procedure of Sample Software

*swdg.c*

```
/* SWDG configuration registers can be written only once. */
/* Before setting, confirm that SWDGn_CFG:LOCK=0*/
if (SWDG0_CFG_LOCK != 0)
{
    return Error;
}

/* Clear reset cause */
SWDG0_PROT = SWDG_PROT_UNLOCK;
SWDG0_RSTCAUSE = 0x00000000;

/* Set the configuration of the interrupt */
SWDG0_PROT = SWDG_PROT_UNLOCK;
SWDG0_INT = unInt.u32Register;

/* Set WDG trigger values */
SWDG0_PROT = SWDG_PROT_UNLOCK;
/* Set the counter clear value*/
SWDG0_TRG0CFG = pstcConfig->u8TriggerKey0;
SWDG0_PROT = SWDG_PROT_UNLOCK;
/* Set the counter clear value */
SWDG0_TRG1CFG = pstcConfig->u8TriggerKey1;

/* Set window lower limit for RUN mode */
SWDG0_PROT = SWDG_PROT_UNLOCK;
SWDG0_RUNLLS = pstcConfig->stcRunModeSettings.u32WindowLowerLimit;

/* Set window upper limit for RUN mode */
SWDG0_PROT = SWDG_PROT_UNLOCK;
SWDG0_RUNULS = pstcConfig->stcRunModeSettings.u32WindowUpperLimit;

/* Set window lower limit for PSS mode */
SWDG0_PROT = SWDG_PROT_UNLOCK;
SWDG0_PSSLLS = pstcConfig->stcPssModeSettings.u32WindowLowerLimit;

/* Set window upper limit for PSS mode */
SWDG0_PROT = SWDG_PROT_UNLOCK;
SWDG0_PSSULS = pstcConfig->stcPssModeSettings.u32WindowUpperLimit;

/* Set delay cycle for Reset or NMI. */
SWDG0_PROT = SWDG_PROT_UNLOCK;
SWDG0_RSTDLY_WDGRSTDLY = pstcConfig->u16ResetDelay;

/* Set the software watchdog timer configuration*/
SWDG0_PROT = SWDG_PROT_UNLOCK;
SWDG0_CFG = unCfg.u32Register;

/* Lock the software watchdog timer configuration */
SWDG0_PROT = SWDG_PROT_UNLOCK;
SWDG0_CFG_LOCK = 1;
```

Write SWDGn_PROT=0 xEDAC_CE55 before performing write access to each register

### 3.1.2 Software Watchdog Timer Clear Procedure

You must write to SWDG0_TRG0 and SWDG0_TRG1 to clear the software watchdog timer counter. SWDG0_TRG0 must match SWDG0_TRG0CFG, and SWDG0_TRG1 must match SWDG0_TRG1CFG.

Figure 3 shows the software watchdog timer clear procedure of the sample software.

Figure 3. Software Watchdog Timer Clear Procedure of Sample Software

*swdg.c*

```
en_result_t Swdg_Clear(uint8_t u8TrgVal0, uint8_t
u8TrgVal1)
{
uint32_t cnt;

/* -------Check parameters and conditions ------- */

cnt = Swdg_GetCounterValue();
if (cnt < SWDG0_RUNLLS )
{
    return Error;
}

/* -------- Access registers -------- */

/* Clear SWDG */
/* Clear counter of software watchdog timer */
IRQ_DISABLE_LOCAL();
SWDG0_TRG0 = u8TrgVal0;
SWDG0_TRG1 = u8TrgVal1;
IRQ_RESTORE();

return Ok;
}
```

## 3.2 Hardware Watchdog Timer Setup Procedure

Table 2 lists the BootROM markers for setting up the hardware watchdog timer. These markers (WDR) are used for this purpose.

If WDR_CEM is enabled, the timer will be started based on the setting defined by the WDR. If WDR_CEM is disabled, the timer will be started based on the default settings in the hardware manual.

Table 2 BootROM Markers for Setting up Hardware Watchdog Timer (WDR)

| Marker Name | Description |
|---|---|
| WDR_INTM | Interrupt configuration marker |
| WDR_TRG0CFGM | Counter clear value setting marker |
| WDR_TRG1CFGM | Counter clear value setting marker |
| WDR_RUNLLM | Lower limit value setting marker in RUN mode |
| WDR_RUNULM | Upper limit value setting marker in RUN mode |
| WDR_PSSLLM | Lower limit value setting marker in PSS mode |
| WDR_PSSULM | Upper limit value setting marker in PSS mode |
| WDR_RSTDLYM | Reset delay counter marker |
| WDR_CFGM | Hardware watchdog timer configuration marker |
| WDR_CEM | Hardware watchdog timer configuration enable marker |

### 3.2.1 Hardware Watchdog Timer Setup Procedure of Sample Software

Figure 4 shows the hardware watchdog timer setup procedure of the sample software.

Figure 4. Hardware Watchdog Timer Setup Procedure of Sample Software

*flash_marker.asm*

```
;*****************************************************************
**
;* 5.4  Watchdog Description Record (WDR)                *
;*****************************************************************
**
;* Hardware Watchdog Interrupt Configuration Marker
;* Reset Enable Marker
;* If set to MARKER_ENABLE, a reset is generated when a watchdog error
occurs.
;* Otherwise, an NMI interrupt is generated in the same condition.
;* < set to MARKER_DISABLE or MARKER_ENABLE
#define WDR_INTM_RSTENM_ENABLE              (MARKER_ENABLE)
;* Prior Warning Interrupt Enable Marker
;* If set to MARKER_ENABLE, prior warning interrupt is enabled.
;* Otherwise, it is disabled.
;* < set to MARKER_DISABLE or MARKER_ENABLE
#define WDR_INTM_IRQENM_ENABLE              (MARKER_ENABLE)

;* Watchdog Trigger 0 Configuration Marker
;* Following values are used to clear hardware watch dog timer.
;* Valid range: 0x00...0xFF
#define WDR_TRG0CFGM_WDGTRG0CFGM_SETTING    (0x00)
#define WDR_TRG1CFGM_WDGTRG0CFGM_SETTING    (0x00)

;* Hardware Watchdog Lower Limit RUN Setting Marker
;* This value defines the lower border of the Watchdog window for RUN
state.
;* When set to "0x00000000", the window function does not work.
#define WDR_WDR_RUNLLM_SETTING              (0x00000000)

;* Hardware Watchdog Upper Limit RUN Setting Marker
;* This value defines the upper border of the Watchdog window for RUN
state.
#define WDR_WDR_RUNULM_SETTING              (0x01000000)

;* Hardware Watchdog Lower Limit PSS Setting Marker
;* This value defines the lower border of the Watchdog window for PSS
state.
;* When set to "0x00000000", the window function does not work.
#define WDR_WDR_PSSLLM_SETTING              (0x00000000)

;* Hardware Watchdog Upper Limit PSS Setting Marker
;* This value defines the upper border of the Watchdog window for PSS
state.
#define WDR_WDR_PSSULM_SETTING              (0x80000000)

;* Hardware Watchdog Reset Delay Counter Marker
;* This value defines the delay to be inserted
;* in Watchdog reset/NMI generation in case of a Hardware Watchdog
error.
;* valid range: 0 to 65535 (0xFFFF)
#define WDR_RSTDLYM_WDGRSTDLYM_SETTING      (0x0000)

;* Clock Selection Marker
;* This value is used to select a source clock of the watchdog counter.
;* < set to WDR_CFGM_CLKSELM_LOW_SPEDD_CR or
WDR_CFGM_CLKSELM_HIGH_SPEDD_CR
#define WDR_CFGM_CLKSELM_CLK_SELECTION
(WDR_CFGM_CLKSELM_LOW_SPEDD_CR)

;* Hardware Watchdog Configuration Enable Marker
;* If set to MARKER_ENABLE, the hardware watchdog is started based
;* on the settings defined with the WDR.
;* Otherwise, the hardware watchdog operates based on the default
settings.
;* < set to MARKER_DISABLE or MARKER_ENABLE
//#define WDR_CEM_ENABLE                      (MARKER_DISABLE)
#define WDR CEM ENABLE                      (MARKER ENABLE)
```

Reset is generated

Prior warning interrupt is enabled

Counter clear value = 0x00

Set the window lower value in RUN mode

Set the window upper value in RUN mode

Set the window lower value in PSS mode (disable)

Set the window upper value in PSS mode

Set the reset delay cycle

Select the source clock (low-speed CR)

WDR setting is enabled for hardware watchdog timer

### 3.2.2 Hardware Watchdog Timer Clear Procedure

WDR_TRG0CFGM and WDR_TRG1CFGM are set by the BootROM marker. You must write to HWDG_TRG0 and HWDG_TRG1 to clear the hardware watchdog timer counter. HWDG_TRG0 must match HWDG_TRG0CFGM, and HWDG_TRG1 must match HWDG_TRG1CFGM.

Figure 5 shows the hardware watchdog timer clear procedure of the sample software.

Figure 5. Hardware Watchdog Timer Clear Procedure of Sample Software

*main.c*

```
int main(void)
{
/* Finalize initialization to default settings. */
/*(this will do IRQ and NMI initialization and global IRQ/NMI enable) */
Start_Init();

/* Endless loop */
for (;;)
    {
        ClearWatchdog();
    }
}

static void ClearWatchdog(void)
{
/* Clear hardware watchdog */
/* Clear counter of hardware watchdog timer */
IRQ_DISABLE_LOCAL();
HWDG_TRG0 = HWDG_TRG0_VALUE;
HWDG_TRG1 = HWDG_TRG1_VALUE;
IRQ_RESTORE();
}
```

# 4 Related Documents

- S6J311E/D/C/B Series Datasheet (Doc. No.002-05681)
- S6J311A/9/8 Series Datasheet (Doc. No.002-04632)
- S6J3110 Series Hardware Manual (Doc.No.002-10667)
- S6J3120 Series Datasheet (Doc.No.002-04863)
- S6J3120 Series Hardware Manual (Doc.No.002-04855)
- S6J3200 Series Datasheet (Doc.No.002-05682)
- S6J3200 Series Hardware Manual (Doc.No.002-04852)
- S6J32E/F/G Series Datasheet (Doc.No.002-10689)
- S6J32E/F/G Series Hardware Manual (Doc.No.002-12500)
- Traveo Family Hardware Manual Platform Part for S6J3200 Series (Doc.No.002-04854)
- S6J3310/20/30/40 Series Datasheet (Doc.No.002-10635)
- S6J3350 Series Datasheet (Doc.No.002-10634)
- S6J3310/20/30/40/50 Series Hardware Manual (Doc.No.002-10185)
- Traveo Family HardwareManual Platform Part for S6J3310/3320/3330/3340/3350 Series (Doc.No.002-07884)
- S6J3360/70 Series Datasheet (Doc.No.002-03359)
- S6J3360/70 Series Hardware Manual (Doc.No.002-18302)
- Traveo Family HardwareManual Platform Part for S6J3360/3370 Series (Doc.No.002-07884)
- S6J3400 Series Datasheet (Doc.No.001-97829)
- S6J3400 Series Hardware Manual (Doc.No.002-09919)
- Traveo Family Hardware Manual Platform Part for S6J3400 Series (Doc.No.002-07884)

## Document History

Document Title: AN204452 - How to Set Up Watchdog Timer for Traveo™ Family

Document Number: 002-04452

| Revision | ECN | Orig. of Change | Submission Date | Description of Change |
|---|---|---|---|---|
| ** | – | KHAS | 06/24/2015 | Initial Release |
| *A | 5041797 | KHAS | 12/09/2015 | Migrated Spansion Application Note from S6J3110_AN708-00012-1v0-E to Cypress format |
| *B | 5187018 | KHAS | 05/11/2016 | Added target products S6J3200/S6J3300/S6J3350 series. |
| *C | 5277823 | HITO | 06/22/2016 | Added target products S6J3400 series |
| *D | 5666824 | HITO | 03/30/2017 | Added target products S6J3360/3370 series |

## 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

### Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

### Technical Support

cypress.com/support



Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709