OpenCPU Watchdog Application Note

GSM/GPRS Module Series

Rev. OpenCPU_Watchdog_Application_Note_V1.0

Date: 2017-07-21

About the Document

History

Revision	Date	Author	Description
1.0	2017-07-10	Chunmao Li	Initial

Contents

Table Index

Figure Index

1 Introduction

This document mainly introduces the OpenCPU watchdog solution, API functions and how to program the watchdog as well.

2 Principle of OpenCPU Watchdog

OpenCPU supports multitasking. Developer may design multitasks to implement the application. To totally prevent each task of the application from getting stuck, OpenCPU has designed an effective watchdog solution that can monitor all user tasks.

2.1. Hardware Reference Design

The following sketch map demonstrates the hardware principle.

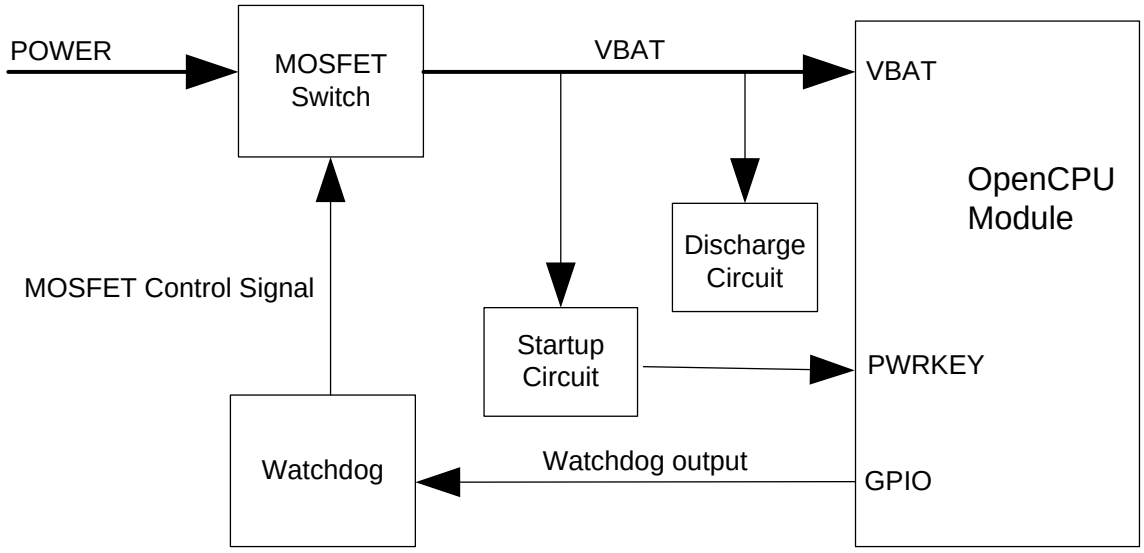

Figure 1: Hardware Design Sketch for Watchdog

The following schematic shows the reference design of hardware.

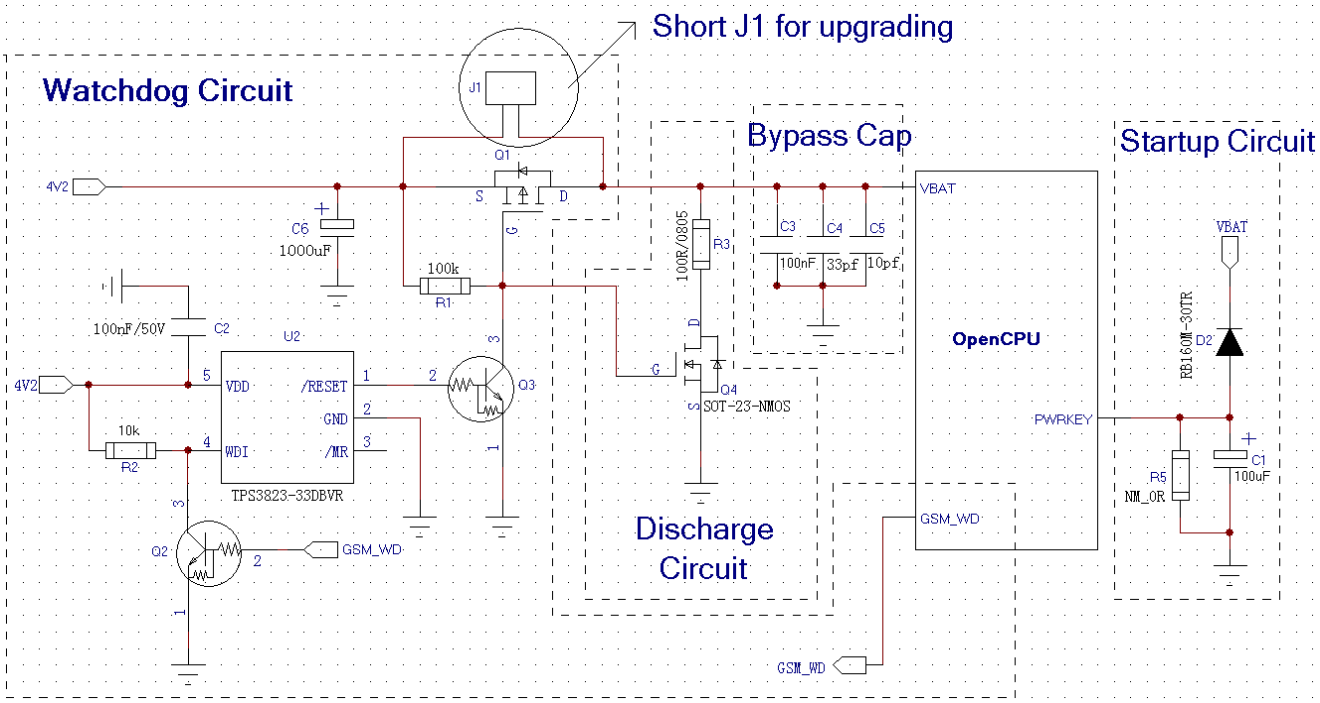

Figure 2: Hardware Reference Design for Watchdog

In the schematic, the watchdog chip “TPS3823-33DBVR” is used. The watchdog chip must have the timeout of at least 1.6 seconds. The following table lists the recommended components.

Table 1: Recommended Components

Location	Part No.	Descriptions	Vendor
U2	TPS3823-33DBVR	IC PROCESSOR SUPERVISORY CIRCUITS SOT23-5 RO	TI
Q1	SI2333CDS-T1-GE3	PMOSFET -12V 4A 45mOHM @VGS=-2.5V SOT23	VISHAY
Q4	SE2306	MOSFET N-Channel Vds=20V Id=6A SOT-23 RO	WILLAS
Q2, Q3	DTC143ZEBTL	NPN 50V 100mA R1=4.7K R2=47K EMT3F RO	Rohm
D2	RB160M-30TR	DIO Schottky 30V 1.0A SOD123 RO	Rohm

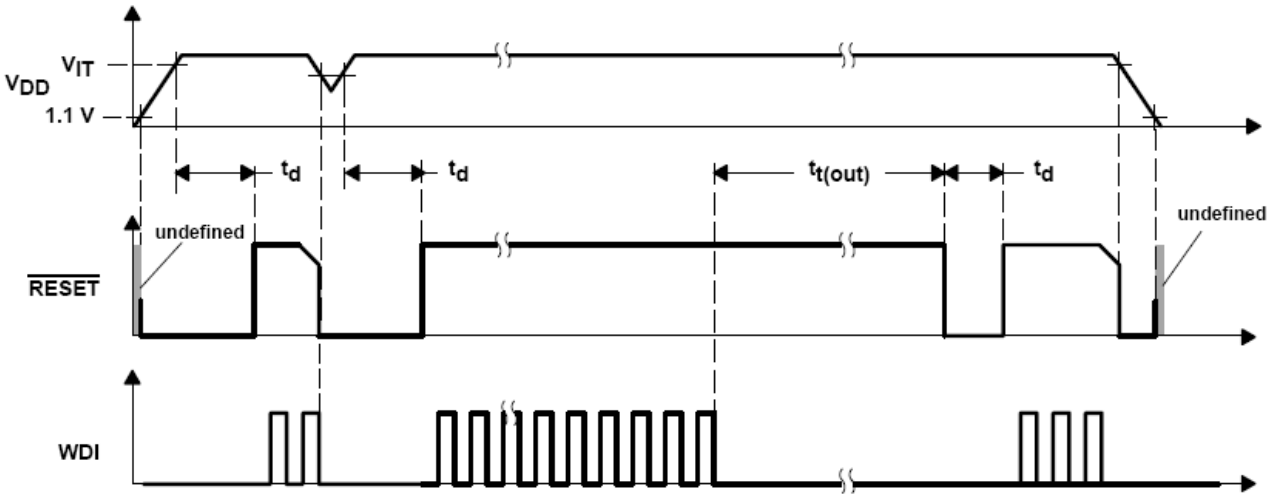

Figure 3: Watchdog Timing Diagram

Combined with Figure 2, when **4V2** is powered on, after VDD becomes higher than V_{IT} and a delay of t_d time, RESET will be pulled high. Then VBAT will supply the module and the module is turned on by startup circuit. Thereafter, the module will output pulse to feed TPS3823-33DBVR and keep RESET high all along, and the TPS3823-33DBVR will monitor VDD all the time. Meanwhile, when VDD drops below the threshold voltage $V_{IT,}$ or TPS3823-33DBVR does not receive pulse during $t_{t(out),}$ RESET will be pulled down.

NOTES

1. The watchdog chip must have the timeout of at least 1.6 seconds.
2. The jumper “J1” will be shorted out when downloading firmware to module on the production line.
3. Pay attention to the position of decoupling capacitor C6, it must be put in front of MOSFET switch Q1, otherwise, it will cause a big dropout in VBAT that triggers watchdog reset.
4. It would be better to feed watchdog in the bootloader, because the time between power on and Operating System start is long and the watchdog will be reset during this time.

2.2. Software Principle

The following sketch map demonstrates the software principle.

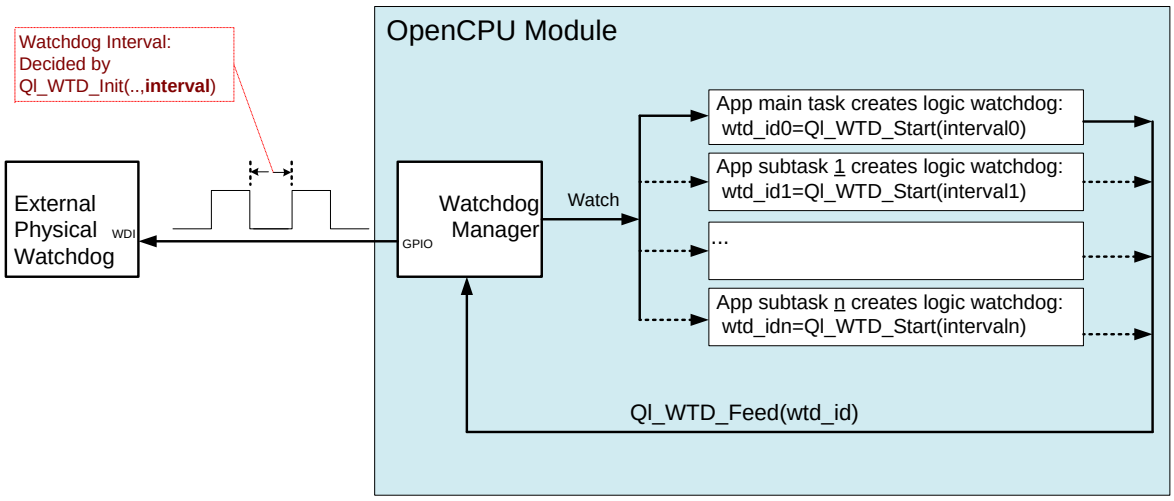

Figure 4: Software Principle for Watchdog Solution

The watchdog solution consists of three primary elements:

- **External Physical Watchdog:** An external watchdog chip, monitors whether the OpenCPU App works normally or not, and it will reset the module when some App task gets stuck. App needs to call **Ql_WTD_Init()** to initialize the external watchdog in any task.
- **Logic Watchdog:** One or more software watchdogs, serves for App tasks. Each task may create a logic watchdog using the API function **Ql_WTD_Start()**.
- **Watchdog Manager:** The software agent of external watchdog, which manages and monitors one or more logic watchdogs and feeds the external watchdog through a GPIO.

The watchdog solution has the following features:

- An external watchdog may monitor all App tasks.
- Customer App does not have to feed the external watchdog, and just needs to simply specify the I/O pin. The watchdog manager automatically feeds the external watchdog. But the customer App tasks have to periodically report its status to watchdog manager by calling **Ql_WTD_Feed()** in each task, in which a logic watchdog has started calling **Ql_WTD_Start()**.
- Each App task may specify the individual overflow time. Developer may specify the different overflow time according to the load of different task.
- When any logic watchdog overflows, the watchdog manager will not feed the external watchdog.

NOTES

1. The interval for feeding external watchdog should be two thirds or less of the overflow-time of the watchdog chip. E.g.: If the overflow-time of the external watchdog is 1.6s, then it would be better when the feeding interval is set to 1s; if the overflow-time of the external watchdog is 3s, then it would be better that the feeding interval is set to 2s.
2. The interval for feeding logic watchdog should be at least twice as long as the interval for feeding external watchdog. The interval for feeding logic watchdog is not that sensitive. You can set it to a long time, such as 30s, 1min, etc. This interval value should be decided according to the load of the task.

2.3. Duty Time and Feed Watchdog

Besides the time that App runs, the external watchdog works in the boot course, App FOTA upgrade course and the production course as well. So the external watchdog has to be fed in whole working period.

The following table lists the working time of the external watchdog and the feeding owner.

Table 2: Owner for Feeding External Watchdog

Period	Owner for Feeding
Booting	Core system
App Running	App
Upgrading App By FOTA	Core system

For the “Booting” and “Upgrading App by FOTA” course, developer just needs to specify the GPIO in “custom_sys_cfg.c” which is designed to connect to the external watchdog.

```
static const ST_ExtWatchdogCfg wtdCfg = {
    PINNAME_GPIO0,     //Specify a pin which connects to the external watchdog, other GPIO is ok.
    PINNAME_END        //Specify another pin for watchdog if needed
};
```

3 API Functions

Developer may call **Ql_WTD_Init()** to initialize the external watchdog in any task, including specifying the I/O pin and feeding interval for external watchdog. The API function **Ql_WTD_Start()** is used to start a logic watchdog, and **Ql_WTD_Feed**() may feed the logic watchdog.

Once App program calls **Ql_WTD_Init()**, it must also call **Ql_WTD_Start**(), or the external watchdog will overflow.

3.1. Ql_WTD_Init

This function initializes watchdog manager, which is responsible for feeding the external watchdog. It can specify the I/O pin and feeding interval.

The interval for feeding external watchdog should be two thirds or less of the overflow-time of the watchdog chip.

- **Prototype**

```
s32 Ql_WTD_Init(s32 resetMode, Enum_PinName wtdPin, u32 interval);
```

- **Parameters**

resetMode:
[in] Must be zero.

wtdPin:
[in] I/O pin that connects to the WDI pin of external watchdog chip.

interval:
[in] The interval for feeding external watchdog. Unit: ms.

- **Return Value**

QL_RET_OK, this function succeeds.
QL_RET_ERR_PARAM, invalid parameter.

3.2. Ql_WTD_Start

The function starts a logic watchdog with the specified interval. If needed, every task may call this function to start a logic watchdog service.

The interval for feeding logic watchdog should be at least twice as long as the interval for feeding external watchdog. The interval for feeding logic watchdog is not that sensitive. You can set it to a long time, such as 30s, 1min, and etc. And the interval value should be decided according to the load of the task.

- **Prototype**

```
s32 Ql_WTD_Start(u32 interval);
```

- **Parameters**

interval:
[in] The interval for feeding the logic watchdog. Unit: ms.

- **Return Value**

This function returns a watchdog ID if succeeds.
QL_RET_ERR_PARAM, invalid parameter.

3.3. Ql_WTD_Feed

This function feeds the logic watchdog which is started by **Ql_WTD_Start()**.

- **Prototype**

```
void Ql_WTD_Feed(s32 wtdID);
```

- **Parameters**

wtdID:
[in] Watchdog ID, which is returned by **Ql_WTD_Start()**.

- **Return Value**

None.

3.4. Ql_WTD_Stop

This function stops the specified logic watchdog.

- **Prototype**

```
void Ql_WTD_Stop(s32 wtdID);
```

- **Parameters**

wtdID:

[in] Watchdog ID, which is returned by **Ql_WTD_Start()**.

- **Return Value**

None.

4 Program Watchdog

The following codes show how to program watchdog in one task and multitasks.

4.1. For Single Task App

```
void proc_main_task(s32 TaskId)
{
    ST_MSG msg;
    s32   wtdId;

    //Init watchdog, GPIO0, 1s interval for external watchdog (suppose that the overflow-timer of external
      watchdog is 1.6s).
    Ql_WTD_Init(1, PINNAME_GPIO0, 1000);

    //Start a logic watchdog service in main task, the max. interval is 5s
    wtdId = Ql_WTD_Start(5*1000);

    //Register & start a timer to feed the logic watchdog.
    Ql_Timer_Register(TIMER_ID_WTD1, callback_onTimer, & wtdId);
    Ql_Timer_Start(TIMER_ID_WTD1, 3000, TRUE);    //The real feeding interval is 3s.

    while (TRUE)
    {
        Ql_memset(&msg, 0x0, sizeof(ST_MSG));
        Ql_OS_GetMessage(&msg);
        switch(msg.message)
        {
        // …
        default:
            break;
        }
    }
}

//Feed the logic watchdog in timer callback
void callback_onTimer(u32 timerId, void* param)
{
```

```
    Ql_WTD_Feed(*((s32*)param));
}
```

4.2. For Multitasks App

Based on the case “one task”, developer just needs to start a new logic watchdog service in multitask, and start a timer to feed the watchdog periodically.

```
void proc_subtask1(s32 TaskId)
{
    ST_MSG msg;
    s32 wtdId;

    //Start a logic watchdog service in subtask, the timeout is 10s.
    wtdId = Ql_WTD_Start(10 * 1000);

    //Register & start a timer to feed the logic watchdog
    Ql_Timer_Register(TIMER_ID_WTD2, callback_onTimer, &wtdId);
    Ql_Timer_Start(TIMER_ID_WTD2, 8 * 1000, TRUE);    //The real interval of feeding watchdog is 8s.

    while (TRUE)
    {
        Ql_memset(&msg, 0x0, sizeof(ST_MSG));
        Ql_OS_GetMessage(&msg);
        switch(msg.message)
        {
            //...
        default:
            break;
        }
    }
}

//Feed the logic watchdog in timer callback
void callback_onTimer(u32 timerId, void* param)
{
    Ql_WTD_Feed(*((s32*)param));
}
```

5 Appendix A Reference

Table 3: Terms and Abbreviations

Abbreviation	Description
App	OpenCPU Application
API	Application Programming Interface
FOTA	Firmware Over The Air
GPIO	General Purpose Input Output
External Watchdog	The external hardware watchdog chip
Watchdog Manager	The software agent of external watchdog, which manages one or more logic watchdogs.
Logic Watchdog	The software watchdog that serves for App task.