FA95/VA95 Non-OS RTC Library Reference Guide

V1.0

Publication Release Date: Oct. 2011

Support Chips:
W55FA Series

Support Platforms:
Non-OS

The information in this document is subject to change without notice.

The Nuvoton Technology Corp. shall not be liable for technical or editorial errors or omissions contained herein; nor for incidental or consequential damages resulting from the furnishing, performance, or use of this material.

This documentation may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from the Nuvoton Technology Corp.

Nuvoton Technology Corp. All rights reserved.

Table of Contents

1. RTC Library

This library is designed to make user application access FA95/VA95 RTC more easily.
The RTC library has the following features:

- Support RTC Current/Alarm time access.
- Support System Power Off Control

1.1. Programming Guide

System Overview

Real Time Clock (RTC) block can be operated by independent power supply while the system power is off. The RTC uses a 32.768 KHz external crystal. It can transmit data to CPU with BCD values. The data includes the time by (second, minute and hour), the day by (day, month and year). In addition, to achieve better frequency accuracy, the RTC counter can be adjusted by software.
The built in RTC is designed to generate the alarm interrupt and periodic interrupt signals. The period interrupt can be 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second. The alarm interrupt indicates that time counter and calendar counter have counted to a specified time recorded in TAR and CAR.
The wakeup signal is used to wake the system up from sleep mode.

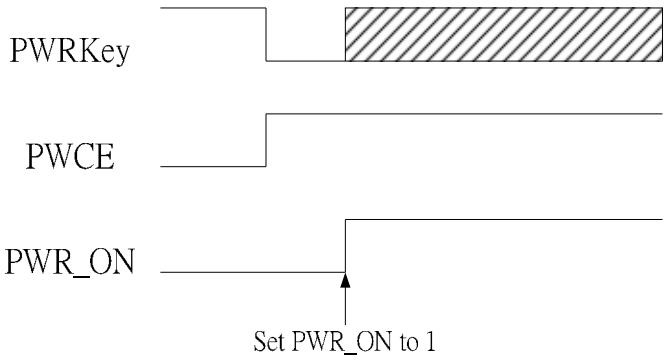
RTC Features

- There is a time counter (second, minute, hour) and calendar counter (day, month, year) for user to check the time.
- Absolute Alarm register (second, minute, hour, day, month, year).
- Relative Alarm
- 12-hour or 24-hour mode is selectable.
- Recognize leap year automatically.
- The day of week counter.
- Frequency compensate register (FCR).
- Beside FCR, all clock and alarm data expressed in BCD code.
- Support time tick interrupt.
- Support wake up function.
- System Power off Control function

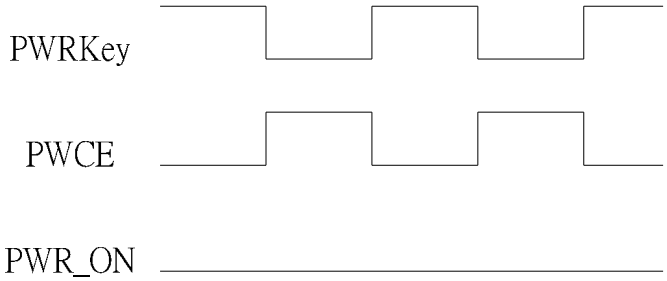
System Power Control Flow

- **Power On from Key**

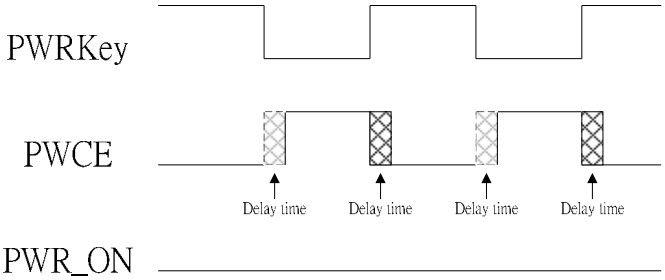
User presses the Power Key to make the Power Control Signal, PWCE to high. If PWR_ON bit is set, the Power Key can be released and the PWCE will be keep on. When system is power on, IBR will set PWR_ON first.



If PWR_ON doesn't be set to 1, the PWCE will back to low when the Power Key is released.



And RTC supports a function (RPWR_DELAY) to postpone the time to set PWCE to high when pressing Power Key or to postpone the time to set PWCE to low when releasing Power Key. The delay time is from 62ms to 868ms. The function is default disabled. User can enable it after first power on and the setting can be kept when RTC is powered even whole system is power-off.



[Note] The function (RPWR_DELAY) only works only when Power Key is pressed or released.

- **Delay Power Control Signal (RPWR_DELAY)**
  The delay time between Power Key pressed and PWCE high (or Power Key released and PWCE low)
  - **3'b000: no change**

- Ø 3'b001: delay 62ms~124ms
- Ø 3'b010: delay 124ms~248ms
- Ø 3'b011: delay 186ms~372ms
- Ø 3'b100: delay 248ms~496ms
- Ø 3'b101: delay 310ms~620ms
- Ø 3'b110: delay 372ms~744ms
- Ø 3'b111: delay 434ms~868ms

n **Normal system Power Control Flow**

The control steps are as follows

1. User press the power key, RPWR, to makes the power control signal, PWCE pin, to high. If the PWR_ON bit, PWRON[0], be set, the power key can be released and the PWCE will keep on. If the PWR_ON bit, PWRON [0], doesn't be set as 1, the PWCE will back to low when the power key is released.
2. If there is another pulse on power key when the PWR_ON bit is set, the system will get an interrupt signal (PSWI). User can decide to clear the PWR_ON or not. If this bit is clear, the PWCE will go to low to turn off the core power. If the PWRON bit is also kept high, the PWCE pin will keep in high level. If there is not any pulse on the power key and the PWR_ON bit is clear by user, the PWCE pin is also set to low at this time.

The follow table is the system power control flow true table.

Input		Output	Note
PWRKey	PWR_ON	PWCE	
X1	X2	Y	
1	0	0	RTC powered only (Default state)
0	0	1	Press key, Power On
0	1	1	keep key & S/W Set X2, Power On
1	1	1	Left key, Power keep On
0	1	1	Press key, get INT, intend to power Off
1	0	0	Left key & S/W clean X2, power Off Or S/W clean X2 , don't need press key, power off
X	1	1	RST_ active, still keep power whenX2=1
PWCE is open drain output **X1, internal pull-up** **X2, it is R/W able** **There is Interrupt from key be pressed**			

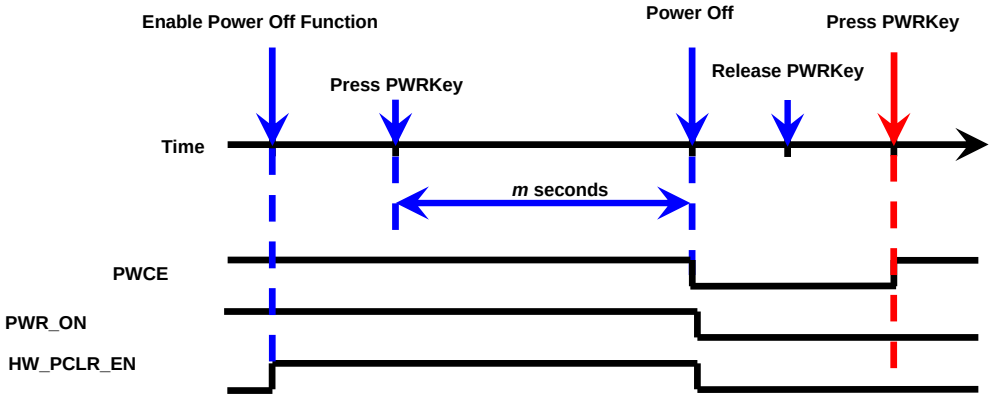
n Force system Power Off Control Flow

The RTC supports a hardware automatic power off function and a software power off function like Notebook. For hardware power off function, it can be enable and disable in HW_PCLR_EN bit and the user presses the power button for a few seconds to power off system. The time to press power the button to power off is configured in PCLR_TIME.

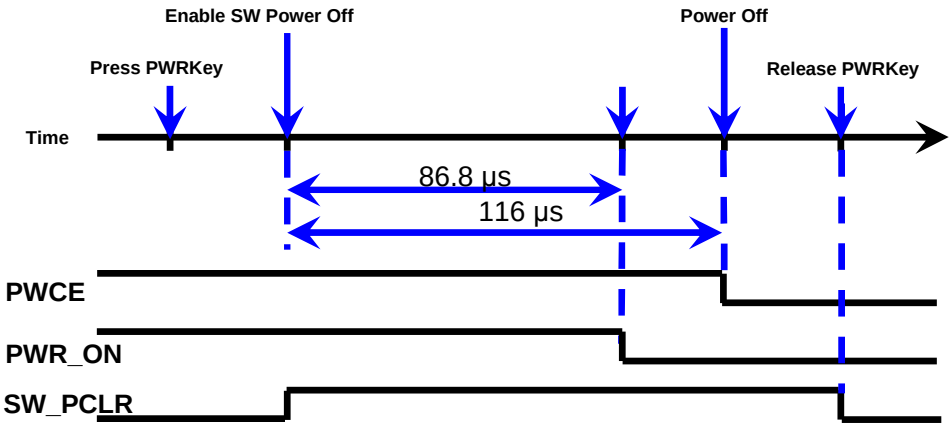
PCLR_TIME Setting	Pressed time to power off	PCLR_TIME Setting	Pressed time to power off
0	Power off right away	8	7~9 seconds
1	0~1 second	9	8~9 seconds
2	1~2 seconds	10	9~10 seconds
3	2~3 seconds	11	10~11 seconds
4	3~4 seconds	12	11~12 seconds
5	4~5 seconds	13	12~13 seconds
6	5~6 seconds	14	13~14 seconds
7	6~7 seconds	15	14~15 seconds

The RTC supports a hardware power off function to provide the power off flow like Notebook. The user presses the power button for a few seconds to power off the system. The time to press power key to power off is counted by hardware. After the time, hardware will set the PWCE to low and clear the PWR_ON and HW_PCLR_EN. After power off, user can decide to set the PWR_ON bit to power on system or not when the PWRKey is pressed.

The timing of the hardware power off function is following



The RTC also supports a software power off function. The user presses the power button for a few seconds to power off the system. The time to press power key to power off is counted by user. When the PWR_ON bit is cleared by user, the PWCE outputs low after 116us and the SW_PCLR bit is cleared when the power key is released. See the timing Figure as following.



RTC Library Constant Definition

Name	Value	Description
RTC_CLOCK_12	0	12-Hour mode
RTC_CLOCK_24	1	24-Hour mode
RTC_AM	1	a.m.
RTC_PM	2	p.m.
RTC_LEAP_YEAR	1	Leap year
RTC_TICK_1_SEC	0	1 tick per second
RTC_TICK_1_2_SEC	1	2 tick per second
RTC_TICK_1_4_SEC	2	4 tick per second
RTC_TICK_1_8_SEC	3	8 tick per second
RTC_TICK_1_16_SEC	4	16 tick per second
RTC_TICK_1_32_SEC	5	32 tick per second
RTC_TICK_1_64_SEC	6	64 tick per second
RTC_TICK_1_128_SEC	7	128 tick per second
RTC_SUNDAY	0	Day of Week: Sunday
RTC_MONDAY	1	Day of Week: Monday
RTC_TUESDAY	2	Day of Week: Tuesday
RTC_WEDNESDAY	3	Day of Week: Wednesday
RTC_THURSDAY	4	Day of Week: Thursday
RTC_FRIDAY	5	Day of Week: Friday
RTC_SATURDAY	6	Day of Week: Saturday
RTC_ALARM_INT	0x01	Aboslute Alarm Interrupt
RTC_TICK_INT	0x02	Tick Interrupt
RTC_PSWI_INT	0x04	Power Switch Interrupt

RTC_RELATIVE_ALARM_INT	0x08	Relative Alarm Interrupt
RTC_ALL_INT	0x0F	All Interrupt
RTC_IOC_IDENTIFY_LEAP_YEAR	0	Identify the leap year command
RTC_IOC_SET_TICK_MODE	1	Set tick mode command
RTC_IOC_GET_TICK	2	Get tick command
RTC_IOC_RESTORE_TICK	3	Restore tick command
RTC_IOC_ENABLE_INT	4	Enable interrupt command
RTC_IOC_DISABLE_INT	5	Disable interrupt command
RTC_IOC_SET_CURRENT_TIME	6	Set Current time command
RTC_IOC_SET_ALAMRM_TIME	7	Set Alarm time command
RTC_IOC_SET_FREQUENCY	8	Set Frequency command
RTC_IOC_SET_POWER_ON	9	Set Power On (Set PWR_ON to 1)
RTC_IOC_SET_POWER_OFF	10	Set Power Off (Set PWR_ON to 0)
RTC_IOC_SET_POWER_OFF_PERIOD	11	Set Power Off Period (PCLR_TIME)
RTC_IOC_ENABLE_HW_POWEROFF	12	Enable H/W Power Off
RTC_IOC_DISABLE_HW_POWEROFF	13	Disable H/W Power Off
RTC_IOC_GET_POWERKEY_STATUS	14	Get Power Key Status
RTC_IOC_SET_PSWI_CALLBACK	15	Set Power Switch Interrupt Callback function
RTC_IOC_GET_SW_STATUS	16	
RTC_IOC_SET_SW_STATUS	17	
RTC_IOC_SET_RELEATIVE_ALARM	18	
RTC_IOC_SET_POWER_KEY_DELAY	19	
RTC_CURRENT_TIME	0	Current time
RTC_ALARM_TIME	1	Alarm time
RTC_WAIT_COUNT	10000	RTC Initial Time out Value
RTC_YEAR2000	2000	RTC Year Reference Value
RTC_RPWR_DELAY_NONE	0	No Delay Power Key (RPWR)
RTC_RPWR_DELAY_62MS	1	Delay Power Key (RPWR) 62ms ~ 124 ms
RTC_RPWR_DELAY_124MS	2	Delay Power Key (RPWR) 124ms ~ 248 ms
RTC_RPWR_DELAY_186MS	3	Delay Power Key (RPWR) 186ms ~ 372 ms
RTC_RPWR_DELAY_248MS	4	Delay Power Key (RPWR) 248ms ~ 496 ms
RTC_RPWR_DELAY_310MS	5	Delay Power Key (RPWR) 310ms ~ 620 ms

RTC_RPWR_DELAY_372MS	6	Delay Power Key (RPWR) 372ms ~ 744 ms
RTC_RPWR_DELAY_434MS	7	Delay Power Key (RPWR) 434ms ~ 868 ms

RTC Library Time and Date Definition

The RTC library provides time structure to access RTC time property.

Name	Value	Description
u8cClockDisplay	RTC_CLOCK_12 / RTC_CLOCK_24	12 Hour Clock / 24 Hour Clock
u8cAmPm	RTC_AM / RTC_PM	the AM hours / the PM hours
u32cSecond	0~59	Second value
u32cMinute	0~59	Minute value
u32cHour	1~11 / 0~23	Hour value
u32cDayOfWeek	RTC_SUNDAY~ RTC_SATURDAY	Day of week
u32cDay	1~31	Day value
u32cMonth	1~12	Month value
u32Year	0~99	Year value

1.2. RTC API

RTC_Init

Synopsis

UINT32 RTC_Init (VOID)

Description

This function is to initialize RTC and install Interrupt service routine

Parameter

None

Return Value

E_SUCCESS - Success

E_RTC_ERR_EIO - Access RTC Failed.

Example

```
/* RTC Initialize */
RTC_Init();
```

RTC_Open

Synopsis

UINT32 RTC_Open (RTC_TIME_DATA_T *sPt)

Description

This function configures RTC current time.

Parameter

sPt	RTC time property and current time information

Return Value

E_SUCCESS	- Success
E_RTC_ERR_EIO	- Access RTC Failed.
E_RTC_ERR_CALENDAR_VALUE	- Wrong Calendar Value
E_RTC_ERR_TIMESACLE_VALUE	- Wrong Time Scale Value
E_RTC_ERR_TIME_VALUE	- Wrong Time Value
E_RTC_ERR_DWR_VALUE	- Wrong Day Value
E_RTC_ERR_FCR_VALUE	- Wrong Compensation value

Example

```
/* Initialization the RTC timer */
if(RTC_Open(&sInitTime) !=E_RTC_SUCCESS)
        sysprintf("Open Fail!!\n");
```

RTC_Close

Synopsis

UINT32 RTC_Close (VOID)

Description

Disable AIC channel of RTC and both tick and alarm interrupt

Parameter

None

Return Value

E_SUCCESS - Success

Example

/* Disable RTC */

RTC_Close();

RTC_Read

Synopsis

UINT32 RTC_Read (E_RTC_TIME_SELECT eTime, RTC_TIME_DATA_T *sPt)

Description

Read current date/time or alarm date/time from RTC

Parameter

eTime The current/alarm time to be read

RTC_CURRENT_TIME - Current time

RTC_ALARM_TIME - Alarm time

sPt RTC time property and current time information

Return Value

E_SUCCESS - Success

E_RTC_ERR_EIO - Access RTC Failed.

E_RTC_ERR_ENOTTY - Command not support, or incorrect parameters.

Example

/* Get the current time */

RTC_Read(RTC_CURRENT_TIME, &sCurTime);

RTC_WriteEnable

Synopsis

UITN32 RTC_WriteEnable (BOOL bEnable)

Description

Enable /Disable RTC register access

Parameter

bEnable TRUE/FALSE

Return Value

E_SUCCESS - Success

E_RTC_ERR_EIO - Access RTC Failed.

Example

```
/* Enable RTC Access */
RTC_WriteEnable(TRUE);
/* Disable RTC Access */
RTC_WriteEnable(FALSE);
```

RTC_DoFrequencyCompensation

Synopsis

BOOL RTC_DoFrequencyCompensation(VOID)

Description

Set Frequency Compensation Data if RTC crystal frequency isn't accurate.

Parameter

None

Return Value

E_SUCCESS - Success

E_RTC_ERR_FCR_VALUE - Can't do compensation.

Example

```
RTC_DoFrequencyCompensation ()
```

RTC_Ioctl

Synopsis

UINT32 RTC_Ioctl (INT32 i32Num, E_RTC_CMD eCmd,, UINT32 u32Arg0, UINT32 u32Arg1)

Description

This function allows user to set some commands for application, the support commands and arguments listed in the table below (Argument 1 is reserved for feature use).

Command	Argument 0	Argument 1	Comment
RTC_IOC_IDENTIFY_LEAP_YEAR	Unsigned integer pointer to store the	None	Get the leap year

	return leap year value		
RTC_IOC_SET_TICK_MODE	Unsigned integer stores the tick mode data	None	Set Tick mode
RTC_IOC_GET_TICK	Unsigned integer pointer to store the return tick number	None	Get the tick counter
RTC_IOC_RESTORE_TICK	None	None	Restore the tick counter
RTC_IOC_ENABLE_INT	interrupt type	None	Enable interrupt
RTC_IOC_DISABLE_INT	interrupt type	None	Disable interrupt
RTC_IOC_SET_CURRENT_TIME	None	None	Set current time
RTC_IOC_SET_ALAMRM_TIME	None	None	Set alarm time
RTC_IOC_SET_FREQUENCY	Unsigned integer stores the Frequency Compensation value	None	Set Frequency Compensation Data
RTC_IOC_SET_PWRON	None	None	Set Power on
RTC_IOC_SET_PWROFF	None	None	Set Power off
RTC_IOC_SET_POWER_OFF_PERIOD	Unsigned integer stores the power off period value : 0~15	None	Set Power Off Period
RTC_IOC_ENABLE_HW_POWEROFF	None	None	Enable H/W Power Off
RTC_IOC_DISABLE_HW_POWEROF	None	None	Disable H/W Power Off
RTC_IOC_GET_POWERKEY_STATUS	Unsigned integer pointer to store the return Power Key status	None	Get Power Key Status
RTC_IOC_SET_PSWI_CALLBACK	The call back function pointer for Power Switch Interrupts	None	Set Power Switch Interrupt Callback function
RTC_IOC_GET_SW_STATUS	Unsigned integer pointer to store the return SW Status	None	Get SW Status data
RTC_IOC_SET_SW_STATUS	Unsigned integer stores the SW Status data	None	Set SW Status data
RTC_IOC_SET_RELEATIVE_ALARM	The call back function pointer for Relative Alarm Interrupts	Alarm time (0~4095)	Set relative alarm and install call backfunvtion
RTC_IOC_SET_POWER_KEY_DELAY	Power Control Selection	None	Set Power Control Signal Delay

Parameter

sicFeature	SIC_SET_CLOCK, SIC_SET_CALLBACK
u32Arg0	Depend on feature setting
u32Arg1	Depend on feature setting

Return Value

None

Example

```
/* Set Tick setting */
RTC_Ioctl(0,RTC_IOC_SET_TICK_MODE, (UINT32)&sTick,0);

* Enable RTC Tick Interrupt and install tick call back function */
RTC_Ioctl(0,RTC_IOC_ENABLE_INT, (UINT32)RTC_TICK_INT,0);

/* Press Power Key during 6 sec to Power off */
RTC_Ioctl(0, RTC_IOC_SET_POWER_OFF_PERIOD, 6, 0);

/* Install the callback function for Power Key Press */
RTC_Ioctl(0, RTC_IOC_SET_PSWI_CALLBACK, (UINT32)PowerKeyPress, 0);

/* Enable Hardware Power off */
RTC_Ioctl(0, RTC_IOC_ENABLE_HW_POWEROFF, 0, 0);

/* Query Power Key Status */
RTC_Ioctl(0, RTC_IOC_GET_POWERKEY_STATUS, (UINT32)&u32PowerKeyStatus, 0);

/* Power Off - S/W can call the API to power off any time he wants */
RTC_Ioctl(0, RTC_IOC_SET_POWER_OFF, 0, 0);

/* Enable RTC Relative alarm Interrupt and install call back function */
RTC_Ioctl(0,RTC_IOC_SET_RELEATIVE_ALARM, 10,
(UINT32)RTC_Releative_AlarmISR);

/* Set Power Control Signal Delay */
RTC_Ioctl(0,RTC_IOC_SET_POWER_KEY_DELAY,    1, 0);
```

[Note 1]. Delay time tabkle

RTC_RPWR_DELAY_NONE**: no change**

RTC_RPWR_DELAY_62MS**: delay 62ms~124ms**

RTC_RPWR_DELAY_124MS**: delay 124ms~248ms**

RTC_RPWR_DELAY_186MS**: delay 186ms~372ms**

RTC_RPWR_DELAY_248MS: **delay 248ms~496ms**

RTC_RPWR_DELAY_310MS: **delay 310ms~620ms**

RTC_RPWR_DELAY_372MS: **delay 372ms~744ms**

RTC_RPWR_DELAY_434MS: **delay 434ms~868ms**

1.3. Example code

The demo code test “Time display”, “Absolute Alarm, “ Relative Alarm” , “Power down Wakeup”, “Software Power Off (Normal Case) Control Flow”, “Hardware Power Off (System Crash) Control Flow”, and “Software Force to Power Off”. Please refer to the RTC sample code of SDK Non-OS.

1.4. Error Code Table

Code Name	Value	Description
E_RTC_SUCCESS	0	Operation success
E_RTC_ERR_CALENDAR_VALUE	1	Wrong Calendar Value
E_RTC_ERR_TIMESACLE_VALUE	2	Wrong Time Scale Value
E_RTC_ERR_TIME_VALUE	3	Wrong Time Value
E_RTC_ERR_DWR_VALUE	4	Wrong Day Value
E_RTC_ERR_FCR_VALUE	5	Wrong Compensation value
E_RTC_ERR_EIO	6	Access RTC Failed.
E_RTC_ERR_ENOTTY	7	Command not support, or parameter incorrect.
E_RTC_ERR_ENODEV	8	Interface number incorrect.

2. Revision History

Version	Date	Description
V1	Oct. 19, 2011	▪ Created

Important Notice

Nuvoton products are not designed, intended, authorized or warranted for use as components in equipment or systems intended for surgical implantation, atomic energy control instruments, aircraft or spacecraft instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for any other applications intended to support or sustain life. Furthermore, Nuvoton products are not intended for applications whereby failure could result or lead to personal injury, death or severe property or environmental damage.

Nuvoton customers using or selling these products for such applications do so at their own risk and agree to fully indemnify Nuvoton for any damages resulting from their improper use or sales.