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.

AN407

A Design Guide to I²C F-RAM™ Processor Companions

Author: Harsha Medu
Associated Part Family: FM3164, FM31256, FM31276, FM31278, FM31L276, and FM31L278
Associated Code Examples: CE211423
Related Application Notes: click here

AN407 provides an overview of and design guidelines for the I²C F-RAM™ real-time clock (RTC) Processor Companion parts: FM3164, FM31256, FM31276, FM31278, FM31L276, and FM31L278. This document also includes a typical application, and an example code.

Contents

1 Introduction

The FM31xx F-RAM product families offer F-RAM memory in 256-Kbit and 64-Kbit densities, with an integrated processor companion and an RTC. The processor companion comprises a power-on system reset, low-voltage detect, a watchdog timer, an early power-fail warning, two event counters, automatic switchover to backup power, and a lockable 64-bit serial number. It employs an industry-standard I²C interface, which is used to access the memory, the processor companion, and the RTC.

The FM3164 and FM31256 operate over a supply voltage range of 2.7 V to 5.5 V, the FM3127x family over 4.0 V to 5.5 V and the FM31L27x family over 2.7 V to 3.6 V. Table 1 summarizes the product families and their major feature sets. The application note AN405 – Comparison between F-RAM I²C Processor Companion Devices describes the differences between each family.

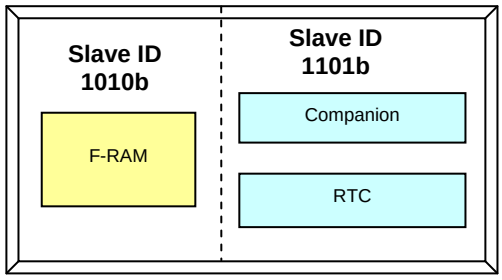
Table 1. Summary of Part Types and Features

Part No.	Memory	Companion	RTC	Voltage	I²C Speed	Package
FM3164	64 Kb	☒	☒	2.7 V–5.5 V	1 MHz	SOIC-14
FM31256	256 Kb	☒	☒			
FM31276	64 Kb	☒	☒	4.0 V–5.5 V	1 MHz	SOIC-14
FM31278	256 Kb	☒	☒			
FM31L276	64 Kb	☒	☒	2.7 V–3.6 V	1 MHz	SOIC-14
FM31L278	256 Kb	☒	☒			

2 Two Logical Devices in One

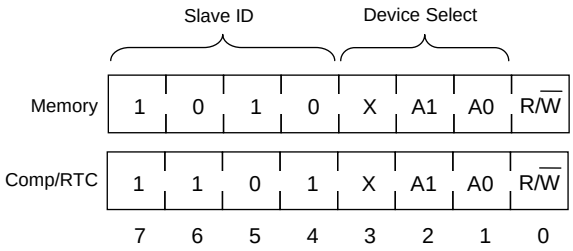
All FM31xx devices are internally organized as two logical devices, as shown in Figure 1: the memory and the companion/RTC. This helps to integrate both the functionalities without affecting each other. Each has their own address space and is accessed via an I²C interface, with each using a unique slave ID: 1010b for the memory and 1101b for the companion/RTC.

Figure 1. Two Logical Devices with Unique Slave IDs



A slave address is required for every transaction in the I²C protocol. The 8-bit slave address is composed of a slave ID, device select bits, and an R/W bit, as shown in Figure 2.

Figure 2. Slave Addresses



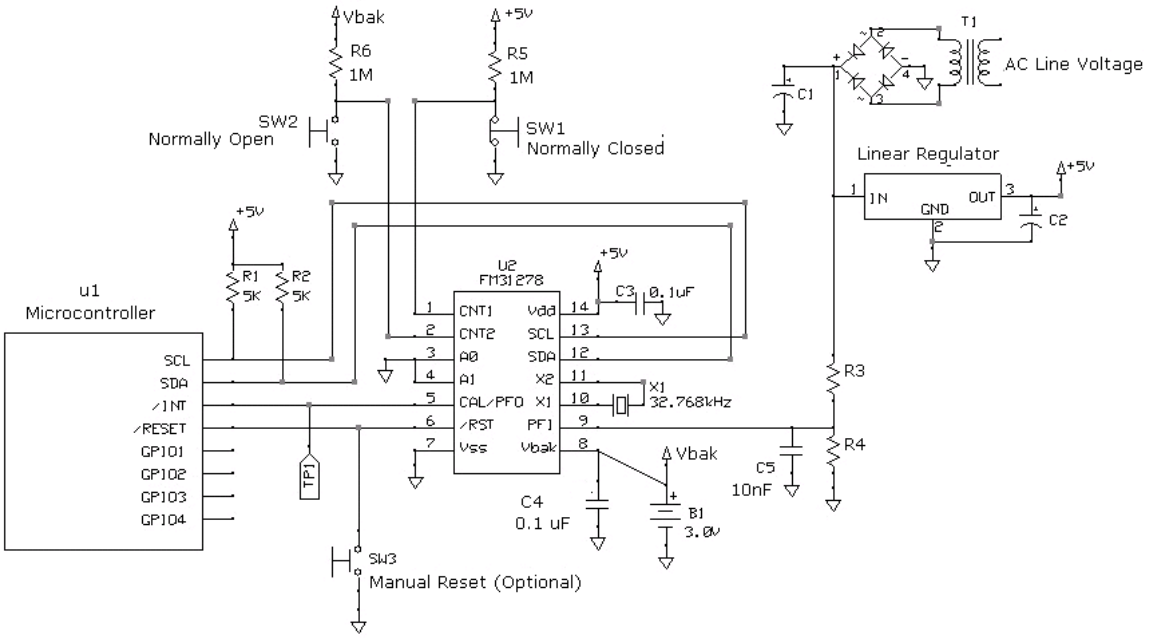
The device select bits A1 and A0 are used to allow multiple devices on the I²C bus to share the same slave ID for memory expansion, yet they are individually addressed. Since the FM31xx devices have two address (device select) pins, up to four devices can share the same I²C bus. This means that you can add three expansion memory devices in addition to an FM31xx device.

3 Typical Application

A typical application of FM31xx (Figure 3) shows external components, their typical values, and connections to other system devices. This application is a line-operated (AC-powered) system with a microcontroller, a FM31278 device, and other passive components. The example microcontroller has a dedicated I²C interface. All microcontrollers may not have this port; in such cases, the I²C protocol may be implemented in firmware and bit-banged through GPIO pins.

Note: Bit banging is a technique used for serial communications. It uses firmware instead of a dedicated hardware. Firmware directly sets and samples the state of pins on the microcontroller, and is responsible for all parameters of the signal like timing, levels, synchronization and so on.

Figure 3. FM3127x Typical Application Circuit



4 Processor Companion Features

The FM31xx family integrates all the necessary processor supervisor features that a designer may need.

The companion features include:

- System power-on reset (POR) with the $\overline{RST}$ pin
  - Low-voltage detect (LVD)
  - Watchdog
  - Manual reset
- Early power-fail warning
- Event counters
- Lockable serial number

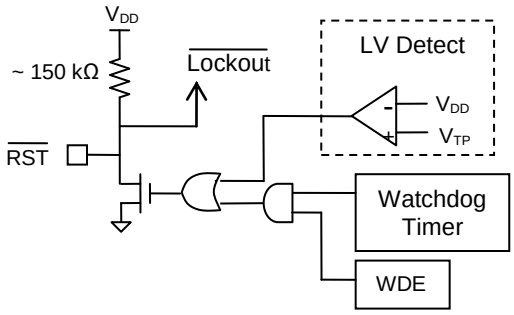
The following sections discuss each of these features in detail.

4.1 System Power-On Reset with $\overline{RST}$ Pin

The FM31xx provides a processor-reset signal when the system powers up and whenever a system fault or manual override (manual hardware reset) occurs. The $\overline{RST}$ pin is primarily used to drive reset to the processor, but can be used as an input to provide a hardware reset to the system. The $\overline{RST}$ pin does not reset or clear any FM31xx internal registers.

There are two trigger sources that can drive reset LOW: a low-voltage (LV) detect circuit and the watchdog timer as shown in Figure 4. At power-up, the $\overline{RST}$ pin is driven LOW as V_{DD} rises toward its nominal operating value. The point at which $\overline{RST}$ is released is determined by the V_{TP} (voltage trip point), an internal trip voltage that is always compared to V_{DD}.

Figure 4. Reset Trigger Sources



An internal pull-up resistor (approximately 150 kΩ) on the $\overline{RST}$ pin eliminates the need for an external resistor. When tripped, the reset circuit times out after approximately 150 ms (100 ms minimum, 200 ms maximum). You may set V_{TP} to one of the following values:

- FM3164/FM31256 devices: 2.6 V, 2.9 V, 3.9 V or 4.4 V (2 bits, VTP1, VTP0 in register 0Bh)
- FM3127x devices: 3.9 V or 4.4 V (VTP in register 0Bh, bit 0)
- FM31L27x devices: 2.6 V or 2.9 V (VTP in register 0Bh, bit 0)

The other source of trigger for the $\overline{RST}$ pin is the watchdog timer, a free-running, user-programmable timer that can be set to time out as soon as 100 ms or as long as 3 seconds. The timer settings and V_{TP} trip voltages are stored in nonvolatile registers, so there is no need to reinitialize these values. For the watchdog timer to trigger a reset, timer must expire and the watchdog enable (WDE = 1, register 0Ah, bit 7) bit must be set. The timer may be restarted at any time within the timeout period by writing 1010b to the Watchdog Restart register (register 09h). To conserve power, the watchdog counter may be disabled by writing 11111b to the watchdog timeout register.

When a reset condition occurs, the device will set one of two flags to indicate the source of the reset in the register 09h: WTR and POR. These bits are battery-backed during power-down. After power is restored, the system host can read these bits to determine the cause of the reset. The host must clear these bits with a separate write instruction after reading them.

Tip During system prototyping and debug, you can clear the WDE bit and still see the watchdog function in action by reading the WTR flag. A watchdog timer value is set, for example, to 01101b or 1.3 seconds. The cleared WDE bit avoids driving the $\overline{RST}$ pin LOW and locking out the I²C interface for a watchdog timer fault, yet you can read the WTR flag in register 09h to see if the software has a problem.

A manual hardware reset may be provided in the system by simply connecting a momentary contact switch to the $\overline{RST}$ pin. The pin detects an external LOW input condition and responds by driving the signal LOW for 200 ms (maximum). This effectively filters and debounces a reset switch. Note that in all reset ($\overline{RST}$) cases, the FM31xx internally blocks all I²C interface traffic and aborts any pending write accesses to the F-RAM array. A reset ($\overline{RST}$) that occurs before all eight data bits and an ACK are received will cause the write to abort. However, write accesses that have been acknowledged by the F-RAM device will be completed, that is, if a reset occurs after the 9th bit of the data phase of the write cycle.

4.2 Early Power-Fail Warning

The F-RAM integrated Processor Companion features a general-purpose comparator that can be used to generate an early power-fail interrupt (PFI). This warning signal can drive a microcontroller interrupt input and must occur before V_{DD} drops too low to save all critical data to the nonvolatile RAM. The Processor Companion's early power-fail interrupt can also be used as a warning to the system to stop conducting critical activity during a brownout condition. For more information, refer to the application note AN400 - Generating a Power-Fail Interrupt using the F-RAM Processor Companion.

4.3 Event Counters

The FM31xx devices integrate two 16-bit event counters for tamper-detect or other event logging purposes. Each counter has an input pin each for the two counters (CNT1 and CNT2), that are edge-triggered and a polarity that is user-defined. The event, or edge, must be a CMOS logic level. The event counter control register and counter values are battery backed.

Figure 3 shows a normally open and a normally closed switch, each connected to ground (or a system case or chassis). The normally open switch is preferred if you do not want extra current pulled from the backup power source. If the system is powered up (V_{DD}) most of the time, an extra few microamperes may not be a problem. On the other hand, an extra few microamperes could reduce the total backup time slightly if you are using a capacitor as your backup source.

At the first power-up with no backup source, the event counters are cleared to all zeroes.

Note: Neither CNT1 nor CNT2 has internal pull-up or pull-down resistors, so they should not be left floating.

4.4 Serial Number

The F-RAM processor companion provides a 64-bit lockable serial number (register 11h to register 18h in Table 2). The serial number provides a unique way of identifying each product. The serial number has unlimited writes until the lock-bit is set by the user.

5 Real-Time Clock

A real-time clock (RTC) counts time in steps of seconds and can report time, day of the week, calendar day, week, and year. It can be programmed to record time when certain system events occur. The RTC consists of an oscillator and counters that derive the time/calendar information, and several registers that hold the time and RTC control settings. The RTC will run continuously even if the main power fails as long as a 3-V backup power source is connected to the V_{BAK} input pin. A backup source can be either a 3-V battery or a super capacitor (see the super capacitor trickle charge discussion in the Backup Power section).

The default factory settings disable the RTC oscillator. To start and configure the RTC, the $\overline{\text{OSCEN}}$ bit (register 01h, bit 7) must first be set to 0. Then, the clock and calendar registers must be written to reflect the current time, day, and date. The RTC register map is shown in Table 2.

Table 2. RTC Register Map

Nonvolatile = ☐ Battery-backed = ☐

Address	Data								Function	Range
	D7	D6	D5	D4	D3	D2	D1	D0		
18h	Serial Number Byte 7								Serial Number 7	FFh
17h	Serial Number Byte 6								Serial Number 6	FFh
16h	Serial Number Byte 5								Serial Number 5	FFh
15h	Serial Number Byte 4								Serial Number 4	FFh
14h	Serial Number Byte 3								Serial Number 3	FFh
13h	Serial Number Byte 2								Serial Number 2	FFh
12h	Serial Number Byte 1								Serial Number 1	FFh
11h	Serial Number Byte 0								Serial Number 0	FFh
10h	Counter 2 MSB								Event Counter 2 MSB	FFh
0Fh	Counter 2 LSB								Event Counter 2 LSB	FFh
0Eh	Counter 1 MSB								Event Counter 1 MSB	FFh
0Dh	Counter 1 LSB								Event Counter 1 LSB	FFh
0Ch					RC	CC	C2P	C1P	Event Count Control	
0Bh	SNL	-	-	WP1	WP0	VBC	VTP1	VTP0	Companion Control	
0Ah	WDE	-	-	WDT4	WDT3	WDT2	WDT1	WDT0	Watchdog Control	
09h	WTR	POR	LB	-	WR3	WR2	WR1	WR0	Watchdog Restart/Flags	
08h	10 years				years				Years	00-99
07h	0	0	0	10 months	months				Month	01-12
06h	0	0	10 date		date				Date	01-31
05h	0	0	0	0	0	day			Day	01-07
04h	0	0	10 hours		hours				Hours	00-23
03h	0	10 minutes			minutes				Minutes	00-59
02h	0	10 seconds			seconds				Seconds	00-59
01h	$\overline{\text{OSCEN}}$	reserved	CALS	CAL4	CAL3	CAL2	CAL1	CAL0	CAL Control	
00h	reserved	CF	reserved	reserved	reserved	CAL	W	R	RTC Control	

The 32.768-kHz oscillator is divided down through a series of counters. The first counter divides it by 32,768 to derive the 1-Hz signal for the second counter. The next counter uses seconds to drive a signal once per minute to the minute counter. Subsequent counters continue to divide down until there is one pulse per hour, month, and year driving their respective counters. The counters hold time and date information that is memory mapped (locations 02h through 08h) into the RTC address space. Because the FM31xx uses a different device ID for the companion register space, they do not appear as F-RAM locations. You may read and write the time and date by reading and writing these locations without affecting the F-RAM locations. The CAL/PFO pin shown in Figure 3 is used to measure the 512-Hz frequency in the calibration mode. This information can be used to correct any offset in the RTC time due to crystal or temperature drift.

In the FM31xx, the RTC data may be read without interrupting the RTC operation. When the RTC is read by writing a '1' to the 'R' bit, a snapshot is taken of the current time-day-date and stored in registers where it can be read by the host. The snapshot ensures that the time doesn't change between successive reads from the host. Similarly, during RTC writes, the registers hold the data written by the host and wait to transfer it to the RTC counters after all the time-day-date information has been written and the 'W' bit is cleared. If W = 0, writes to the RTC registers are ignored. For more information on RTC read and write, see Setup Example.

5.1 Backup Power

The RTC, Event Counter, and various control settings are maintained by using a backup source on the V_{BAK} pin even if the primary power source is removed. The V_{BAK} source may be a battery or a large-value super capacitor. For more information, refer to AN404 - F-RAM RTC Backup Supply (V_{BAK} pin) and UL Compliance.

When using a super capacitor as the backup source, the capacitor can discharge over a long period of time to a point where the V_{BAK} voltage drops out of spec and the RTC stops functioning, resulting in the loss of backup data. To avoid this situation, the charge on the capacitor needs to be restored. For different types of charging, refer to AN401 - Charging Methods for the F-RAM RTC Backup Capacitor.

5.2 RTC Calibration

RTC calibration is required primarily to compensate for the frequency shift due to crystal tolerance, temperate effect, and load capacitance mismatch. A 512-Hz signal brought out on the CAL/PFO pin is used for calibration purposes. For more information about calibration, refer to AN402 - F-RAM RTC Oscillator Design Guide.

5.3 Setup Example

The following example is a setup procedure that describes the proper sequence for the power-up, initialization, and register settings.

1. Apply V_{DD} power.
2. Apply battery to V_{BAK}. If a super capacitor is used, you can set the trickle charger (VBC = 1) and optionally the Fast Charge (FC) bit. If a battery is used, ensure that VBC = 0.
3. Set the V_{DD} voltage trip point bit, VTP, if you prefer a setting higher than the default setting.
4. Enable the RTC oscillator (set $\overline{\text{OSCEN}}$ = 0).
5. Enter the RTC calibration mode (CAL = 1).
6. Determine the RTC clock error and sign by monitoring the 512-Hz output (CAL/PFO pin). Use a frequency counter.
7. Write the error correction value to CALS and CAL (4:0). See the calibration adjustments table in the datasheet.
8. Exit the calibration mode (CAL = 0).
9. Set the Write bit (W bit) to enable writing the time-day-date values to the RTC.
10. Write the time-day-date values to registers 02h–08h.
11. Clear the W bit to start the RTC with new values.
12. Resume Normal operation.

To read the RTC, follow this simple three-step procedure:

1. Set the Read bit (R bit), which takes a snapshot of the RTC registers (assumes R is previously at logic 0).
2. Issue the Read Processor Companion command (slave ID 1101b), starting at address 02h, and read seven RTC bytes (02h–08h).
3. Clear the R bit to prepare for the next RTC read.

6 Summary

This application note summarizes the features of the FM31xx family of devices. This document also provides a typical application, design guidelines, and example pseudo codes for the FM31xx devices.

7 Related Application Notes

- AN400 - Generating a Power-Fail Interrupt using the F-RAM Processor Companion
- AN401 - Charging Methods for the F-RAM RTC Backup Capacitor
- AN402 - F-RAM RTC Oscillator Design Guide
- AN404 - F-RAM RTC Backup Supply (V_{BAK} pin) and UL Compliance
- AN408 - A Design Guide to SPI F-RAM Processor Companion - FM33256B

8 Datasheets

- FM3164/FM31256: 64-Kbit/256-Kbit Integrated Processor Companion with F-RAM
- FM31276/FM31278: 64-Kbit/256-Kbit Integrated Processor Companion with F-RAM
- FM31L276/FM31L278: 64-Kbit/256-Kbit Integrated Processor Companion with F-RAM

9 PSoC 3-Based Application Code Examples

9.1 Enable RTC Oscillator

```
        /************************* Enable RTC Oscillator
        ******************************/
        data[0] = 0x00;    // Data for clearing the OSCEN bit

        WRITE_RTC (0xD0,      // 0xD0 is the command for a write to the Companion/RTC
                   0x01,      // Sets the address pointer to Register 01h
                   data,  // Writes data 0x00 which clears the OSCEN bit
                   0x01); // Number of bytes to be written
```

9.2 Set RTC Time/Date

```
        /************************* Set RTC time/date ******************************/

        // Step #1 Set W bit which allows writes to RTC registers
        data[0] = 0x02;    // Data for setting the W bit

        WRITE_RTC (0xD0,      // 0xD0 is the command for a write to the Companion/RTC
                   0x00,      // Sets the address pointer to Register 00h
                   data,  // Writes data 0x02 which sets the W bit
                   0x01); // Number of bytes to be written

        // Step #2 Write time/date to RTC Registers
        data[0] = 0x00;    // Seconds set to 00
        data[1] = 0x10;    // Minutes set to 10
        data[2] = 0x14;    // Hours set to 14 (2 PM)
        data[3] = 0x03;    // Day set to the third day of the week
        data[4] = 0x04;    // Date set to the fourth day in March
        data[5] = 0x03;    // Month set to March
        data[6] = 0x08;    // Year set to 2008

WRITE_RTC (0xD0,           // 0xD0 is the command for a write to the Companion/RTC
           0x02,           // Sets the address pointer to Register 02h
                   data,               // Writes time/date
           0x07);          // Number of bytes to be written

        // RTC does not start to run yet.

        // Step #3 Clear W bit to start RTC with the exact time
        data[0] = 0x00;            // data=0x00 clears the W bit

WRITE_RTC (0xD0,           // 0xD0 is the command for a write to the Companion
           0x00,           // Sets the address pointer to Register 00h
           data,           // Data=0x00 clears the W bit to start RTC with time
defined
                           // in Step #2. The 8th clock of data byte defines the
actual
                           // start of the RTC.
           0x01);          // Number of bytes to be written
```

9.3 Set VTP Voltage Detect Trip Point

```
      /********************* Set VTP Voltage Detect Trip Point
      *************************/
      // From the factory, VTP bit is cleared to 0. If user wants to set trip point
      to
      // the higher setting, then write VTP bit to 1 in Reg 0Bh control register.

      data[0] = 0x01;   // Data for setting VTP bit

      WRITE_RTC (0xD0,     // 0xD0 is the command for a write to the Companion/RTC
                 0x0B,     // Sets the address pointer to Register 0Bh
                 data,  // Writes data 0x01 which sets the VTP bit
                 0x01); // Number of bytes to be written
```

9.4 Read RTC Registers

```
      /************************* Read RTC Registers ******************************/

      // Step #1 Set R bit which takes snapshot of RTC registers
      data[0] = 0x01;   // Data for setting the R bit

      WRITE_RTC (0xD0,     // 0xD0 is the command for a write to the Companion/RTC
                 0x00,     // Sets the address pointer to Register 00h
                 data,  // Writes data 0x01 which sets the R bit
                 0x01); // Number of bytes to be written

      // Step #2 Read RTC Registers
READ_RTC (0xD0,           // 0xD0 is the command for a write to the Companion/RTC
          0x02,           // Sets the address pointer to Register 02h
                data,             // Data buffer to read RTC registers
          0x07);          // Number of bytes to be read

      // data buffer contains the following
      // 0x59 - Seconds
      // 0x15 - Minute
      // 0x14 - Hour
      // 0x03 - Third day of the week
      // 0x04 - Fourth day in March
      // 0x03 - March
      // 0x08 - 2008

      // Step #3 Clear R bit
      data[0] = 0x00;           // data=0x00 clears the W bit

WRITE_RTC (0xD0,          // 0xD0 is the command for a write to the Companion
           0x00,          // Sets the address pointer to Register 00h
           data,          // Data=0x00 clears the R bit to allow RTC read next time
           0x01);         // Number of bytes to be written
```

9.5 Calibrate RTC

```
      /************************* Calibrate RTC ******************************/

      // Step #1 Set CAL bit which turns on the 512Hz sq wave on the CAL/PFO pin.
      Setting
      // the CAL bit also allows writes to NV bits in Reg 01h CAL/Control register.
      data[0] = 0x04;   // Data for setting the CAL bit

      WRITE_RTC (0xD0,     // 0xD0 is the command for a write to the Companion/RTC
                 0x00,     // Sets the address pointer to Register 00h
                 data,  // Writes data 0x04 which sets the CAL bit
                 0x01); // Number of bytes to be written

      // Step #2 Use a freq counter to accurately measure the 512Hz, use lookup table
      in
      // datasheet to find to that matches your freq measurement, write cal code to
      Reg 01h
      data[0] = 0x2C;   // Data=0x2C for 511.9722Hz and 511.9744Hz

WRITE_RTC (0xD0,           // 0xD0 is the command for a write to the Companion/RTC
           0x01,           // Sets the address pointer to Register 01h
                  data,              // Writes calibration value 0x2C
           0x01);          // Number of bytes to be written

      // Step #3 Clear CAL bit to exit cal mode and turn off 512Hz
      data[0] = 0x00;            // data=0x00 clears the CAL bit

WRITE_RTC (0xD0,           // 0xD0 is the command for a write to the Companion
           0x00,           // Sets the address pointer to Register 00h
           data,           // Data=0x00 clears the CAL bit
           0x01);          // Number of bytes to be written
```

9.6 Configure Event Counters

```
      /************************* Configure Event Counters
      ******************************/

      // Configure polarity bits for rising edge detection on Counter1 and falling
      edge
      // detection on Counter2
      data[0] = 0x01;   // Data for setting C2P to 0, C1P to 1

      WRITE_RTC (0xD0,     // 0xD0 is the command for a write to the Companion/RTC
                 0x0C,     // Sets the address pointer to Register 0Ch
                 data,  // Writes data 0x01 to set C2P to 0, C1P to 1
                 0x01); // Number of bytes to be written
```

9.7 Read Event Counters

```
      /************************* Read Event Counters
      ******************************/

      // Step #1 Set RC bit which takes snapshot of both counter registers
      data[0] = 0x09;   // Data for setting the RC bit and keeps C1P=1

      WRITE_RTC (0xD0,     // 0xD0 is the command for a write to the Companion/RTC
                 0x0C,     // Sets the address pointer to Register 0Ch
                 data,  // Writes data 0x09 which sets the R bit
                 0x01); // Number of bytes to be written

      // Step #2 Read Event Counters
READ_RTC (0xD0,            // 0xD0 is the command for a write to the Companion/RTC
          0x0D,            // Sets the address pointer to Register 0Dh
                data,               // Data buffer to read RTC registers
          0x04);           // Number of bytes to be read

      // data buffer contains the following
      // 0x1A - Counter1 reads out LSB 0x1A (decimal 26)
      // 0x00 - Counter1 reads out MSB 0x00
      // 0x31 - Counter2 reads out LSB 0x31 (decimal 49)
      // 0x02 - Counter2 reads out MSB 0x02

      // Step #3 Clear RC bit
      data[0] = 0x01;            // data=0x01 clears the RC bit

WRITE_RTC (0xD0,           // 0xD0 is the command for a write to the Companion
           0x0C,           // Sets the address pointer to Register 0Ch
           data,           // Data=0x01 clears the RC bit and keeps C1P=1
           0x01);          // Number of bytes to be written
```

9.8 I²C Processor Companion Write

```
/*****************PSoC3 Based Pseudo Code for I2C Processor Companion
Write**************/
uint8 WRITE_RTC (uint8 slave_id, uint8 addr, uint8 *data_write_ptr, uint8
total_data_count )
{
    uint32 i;
    uint8 error_status;

    // I2C Start Condition
    error_status = nvRAM_I2C_1_I2C_MasterSendStart(slave_id, 0);
    if( error_status != nvRAM_I2C_1_I2C_MSTR_NO_ERROR )
    {
        nvRAM_I2C_1_I2C_MasterSendStop();
       return error_status;
    }

    // Send F-RAM Processor Companion Register Address
   error_status = nvRAM_I2C_1_I2C_MasterWriteByte((uint8)(addr));
    if( error_status != nvRAM_I2C_1_I2C_MSTR_NO_ERROR )
    {
        nvRAM_I2C_1_I2C_MasterSendStop();
       return error_status;
    }

   // Send F-RAM Processor Companion Register Data Bytes
   for(i = 0; i < total_data_count; i++ )
   {
     error_status =
nvRAM_I2C_1_I2C_MasterWriteByte((uint8)(data_write_ptr[i]));
      if( error_status != nvRAM_I2C_1_I2C_MSTR_NO_ERROR)
      {
          nvRAM_I2C_1_I2C_MasterSendStop();
         return error_status;
      }
   }

   // I2C Stop Condition
   nvRAM_I2C_1_I2C_MasterSendStop();

   return error_status;
}
```

9.9 I²C Processor Companion Read

```
/*****************PSoC3 Based Pseudo Code for I2C Processor Companion
Read***************/
uint8 READ_RTC (uint8 slave_id, uint8 addr, uint8 *data_read_ptr, uint8
total_data_count)
{
    uint32 i;
    uint8 error_status;

    // I2C Start Condition
    error_status = nvRAM_I2C_1_I2C_MasterSendStart(slave_id, 0);
    if( error_status != nvRAM_I2C_1_I2C_MSTR_NO_ERROR )
    {
        nvRAM_I2C_1_I2C_MasterSendStop();
        return error_status;
    }

    // Send F-RAM Processor Companion Register Address
    error_status = nvRAM_I2C_1_I2C_MasterWriteByte((uint8)(addr));
    if( error_status != nvRAM_I2C_1_I2C_MSTR_NO_ERROR )
    {
        nvRAM_I2C_1_I2C_MasterSendStop();
        return error_status;
    }

    // I2C Restart Condition
    error_status = nvRAM_I2C_1_I2C_MasterSendRestart(slave_id, 1);
    if( error_status != nvRAM_I2C_1_I2C_MSTR_NO_ERROR )
    {
        nvRAM_I2C_1_I2C_MasterSendStop();
        return error_status;
    }

    // Read F-RAM Processor Companion Register Data Bytes
    for(i = 0; i < total_data_count - 1; i++ )
    {
       data_read_ptr[i] = nvRAM_I2C_1_I2C_MasterReadByte(1);
    }

    // Read is stopped by NACKing the last byte
    data_read_ptr[i] = nvRAM_I2C_1_I2C_MasterReadByte(0);

    // I2C Stop Condition
    nvRAM_I2C_1_I2C_MasterSendStop();

    return error_status;
}
```

Document History

Document Title: AN407 - A Design Guide to I²C F-RAM™ Processor Companions

Document Number: 001-87421

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	4039506	MEDU	06/25/2013	New Spec.
*A	4573031	MEDU	11/27/2014	Updated Document Title to read as “A Design Guide to I²C F-RAM™ Processor Companions – AN407”. Updated Associated Project as “Yes” in page 1. Updated Associated Part Family as “FM3164, FM31256, FM31276, FM31278, FM31L276, and FM31L278” in page 1. Updated Abstract: Updated description. Updated Introduction: Renamed “Overview” as “Introduction” in heading. Updated description and Table 1. Updated Two Logical Devices in One: Updated description and Figure 1. Updated Typical Application: Updated description. Updated Processor Companion Features: Updated description. Updated Real-Time Clock: Updated description. Updated Backup Power: Updated description. Removed “Calculating Capacitor Backup Time”. Added RTC Calibration. Updated Setup Example: Updated description. Added Related Application Notes. Added Datasheets. Removed “Pseudo Code Examples”. Added PSoC 3-Based Application Code Examples. Added PSoC 3 based user module project as attachment.
*B	5293268	MEDU	06/02/2016	Added a reference to code example CE211423 Updated template
*C	5826215	AESATMP9	07/20/2017	Updated logo and copyright.

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

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