Author: J. Pflasterer

ISL12022M EEPROM Recall Procedure

Register Structure

The ISL12022M device is a high-accuracy, temperature compensated Real Time Clock (RTC). An internal temperature sensor and state machine adjust the oscillator frequency to keep error less than ±5ppm. Most of the registers in the device are RAM-based, read and writable by a microcontroller. There are also compensation registers which have a special setup.

The compensation registers are EEPROM-based, and recalled into RAM for use by the state machine. They are programmable only during factory test, no other programming of these EEPROM registers is possible. The RAM mirror registers are accessible, however, with two different types. One type is readable by the user during normal operation, but not writable. These are in the control registers and contain initial calibration values for the crystal oscillator frequency compensation. The other type of register is for internal test and calibration of voltage references, A/D converter, and temperature sensor. These registers are locked and are normally not accessible by the user.

Power-up EEPROM Recall

The ISL12022M has V_{DD} and V_{BAT} power inputs. Either input may have power applied first, and this will trigger the transfer of the EEPROM contents to the RAM (called Power On Recall or POR). The POR normally proceeds with no issues and the device will function per the data sheet, with the serial interface communicating normally. Note that in systems with no battery, and V_{BAT} grounded, the V_{DD} pin alone will determine POR quality.

There are special situations which can affect proper POR. The most frequent problems involve power supply ramp speed or untimely glitches during power-up. Like most EEPROM-based devices, a long slow V_{DD} ramp will trigger recall at too low a voltage and the RAM registers will have random or improper data. Ramp rates slower than 50mV/ms can possibly cause failed POR. Note that this is not listed in the data sheet, but is extremely slow ramp and does not practically occur in most systems.

Power glitches, especially in the region of about 1.5V to 2.0V, can also trigger a failed POR, depending on the speed and amplitude of the glitch. A well filtered power supply should prevent glitches, and local decoupling at the V_{DD} pin will help prevent any problems. If glitches are common in a system, a local R-C filter at the V_{DD} pin will help greatly.

Note that the data sheet lists the maximum V_{DD} *Negative* power down ramp at -10V/ms. This only applies to circuits using the V_{BAT} pin for backup power, whereby the fast power down ramp does not allow the V_{DD}/V_{BAT} switch enough time to switch to the battery input while the V_{DD} voltage has dropped below that required to keep the RAM intact. This can also cause RAM corruption, and should be avoided.

EEPROM Recall Verification and Repair

In the case where a failed POR cannot be anticipated or avoided (such as a lightning event), it is possible to perform a procedure to place the correct contents in the calibration RAM, by unlocking the internal test and calibration registers. These registers are normally inaccessible to a user, but there is one register which allows unlocking them for reading and writing. The EEPROM locations can be read (only) and the contents used to overwrite the test and calibration RAM registers.

The flowchart in Figure 1 outlines a procedure which reads the EEPROM contents and compares to the recalled RAM contents, and overwrites the RAM contents only if there is a discrepancy between RAM and EEPROM. This procedure has been used and verified to work at restoring corrupted registers.

The actual procedure is as follows. This can be used to write microcontroller code for a power-up routine to test for failed recall and correct, if necessary. Description is in parenthesis.

1. Write 03h to address FFh. (Sets the device into test mode to access EEPROM (read only) and RAM.)
2. Write data 00h to addresses 48h through 4Bh (4 bytes). (Clears test registers and sets device into normal mode, not any test modes.)
3. Read data from addresses 30h to 37h (8 bytes, 1 page), store that data. RTC EEPROM to MCU RAM (these are the main calibration and setup registers. Read data from the EEPROM and bypass the RAM or functional registers).
4. Read data from addresses 40h to 47h (8 bytes, 1 page), store that data. RTC EEPROM to MCU RAM (these are also calibration and setup registers. Read data from the EEPROM and bypass the RAM or functional registers).
5. Write 01h to address FFh. (Sets the device into test mode to access RAM only.)
6. Read data from addresses 30h to 37h (8 bytes, 1 page), store that data. RTC RAM to MCU RAM (these are the RAM calibration registers with the recalled data).
7. Read data from addresses 40h to 47h (8 bytes, 1 page), store that data. RTC RAM to MCU RAM (these are also RAM calibration registers with the recalled data).
8. Read data from addresses 0Bh, 0Ch, 0Dh (3 bytes only), store that data. RTC RAM to MCU RAM (these are RAM control registers with the recalled data, which are the same as addresses 30h, 31h, 32h).
9. Read data from addresses 2Ch, 2Dh (2 bytes only), store that data. RTC RAM to MCU RAM (these are RAM control registers with the recalled data, which are the same as addresses 33h, 42h).
10. Compare stored EEPROM data with the stored RAM data. Each page and byte is compared to check for any bits recalled incorrectly. If there is an incorrect bit anywhere, begin overwrite routine.

11. Write the data stored from EEPROM 30h-37h to device addresses 30h-37h (with RAM test mode enabled, all writes go to RAM and will overwrite the existing recalled data. You will NOT overwrite the EEPROM, that takes a complicated sequence with high voltage).
12. Write the data stored from EEPROM 40h-47h to device addresses 40h-47h. (overwrite RAM).
13. Read only register 34h, store (1 byte). (Read from existing MCU RAM and store 1 byte to MCU RAM. Need to set the override bit for writing directly to the device control registers. These registers are identical to the recalled calibration registers, cannot normally be written by the user, and need to be set correctly.)
    1. Set bit 6 to "0" (that is, mask off all bits and make bit 6 a "0" with all other bits the same).
    2. Write the new data back to register 34h.
    3. Write data read from address 30h to register 0Bh. From MCU RAM to RTC RAM
    4. Write data read from address 31h to register 0Ch. From MCU RAM to RTC RAM
    5. Write data read from address 32h to register 0Dh. From MCU RAM to RTC RAM
    6. Write data read from address 33h to register 2Ch. From MCU RAM to RTC RAM
    7. Write data read from address 42h to register 2Dh. From MCU RAM to RTC RAM
14. Write original data from register 34h back to register 34h (with bit 6 back to "0"). MCU RAM to RTC RAM
15. Write 00h to address FFh. RTC RAM; disables test mode, essentially locking the calibration RAM contents until power is lost.

Sample C Code (Pseudocode)

Following is sample C code for the POR recall repair. This code is intended to be used at the first power-up. It can also be used If a power supply glitch is suspected or as a periodic check/repair utility in the microcode. This code is not intended to be for a specific microcontroller or as a complete code solution for the ISL12022M device.

```
/*********************************************************
* This code is intended to be pseudo C code to aid in *
* the development of firmware for the Intersil RTC, with*
* part number ISL12022M. Some functions are declared but*
* not implemented as different uC architectures handle*
* these functions differently.*
*********************************************************/
//Unimplemented utility functions
/*Format:
Function name
Pseudo Code declaration
Description of arguments and return values
*/

//Write I2C
void write_I2C(data, addr, numBytes);
//data is data to write (variable size)
//address is the starting address
//numBytes is the number of bytes to write. Matches size of "data"

//Read I2C
array read_I2C(addr, numBytes);
//returns an array of size "numBytes" which contains the read data starting at address 'addr'

//Compare
boolean compare(src, dest)
//Returns the truth value of src==dest
```

```
//"Main Loop"

function start_up()
{

//Enter test Mode
write_I2C(0x03, 0xff, 1)

//Clear RAM test Config Reg
write_I2C([0x00,0x00,0x00,0x00],0x48,4)

//Transfer EEPROM to local storage
MCU_A1 = read_I2C(0x30,8)
MCU_A2 = read_I2C(0x40,8)

//Enter RAM test Mode
write_I2C(0x01,0xff,1)

//Transfer recalled RAM to local storage

MCU_B1 = read_I2C(0x30,8)
MCU_B2 = read_I2C(0x40,8)
MCU_B3 = [read_I2C(0x0B,1),read_I2C(0x0C,1),read_I2C(0x0D,1)] //array of 3 Bytes
MCU_B4 = [read_I2C(0x2C,1),read_I2C(0x2D,1)] //array of 2 Bytes

//Check

if(compare(MCU_A1,MCU_B1) && compare(MCU_A2,MCU_B2) && compare(MCU_A1[1:3],MCU_B3) &&
compare(MCU_A1[4],MCU_B4[1]) && compare(MCU_A2[3],MCU_B4[2]) )
write_I2C(0x00,0xFF,1) //exit test mode
else
fixErrors()

}

//Error Fix

function fixErrors(){

I2C_write(MCU_A1, 0x30,8) //write A1 to RTC_RAM[0x40]
I2C_write(MCU_A2, 0x40,8) //write A2 to RTC_RAM[0x40]
MCU_D1 = MCU_A1[5] //Cop byte 5 of A1 to D1

MCU_D2 = (MCU_D1 & 0xBF) //0xBF is a bitmask, turns off bit 6

I2C_write(MCU_D2, 0x34, 1) //write D2 to RTC_RAM[0x34]

I2C_write(MCU_A1[1] 0x0B,1)//write first 3 bytes of A1 to RTC_RAM[0x0b-0c]
I2C_write(MCU_A1[2] 0x0C,1)
I2C_write(MCU_A1[3] 0x0D,1)
```

```
I2C_write(MCU_A1[4], 0x2C,1) //write A1[4] to RTC_RAM[0x2c]
I2C_write(MCU_A2[3] 0x2D,1) //write A2[3] to RTC_RAM[0x2d]

I2C_write(MCU_D1, 0x34, 1) //write D1 to RTC_RAM[0x34]

write_I2C(0x00,0xFF,1) //exit test mode

}
*****************************
End of C code.
```

Summary

The ISL12022M provides a high accuracy 3-in-1 RTC solution for timing requirements. Care must be taken with the V_{DD} supply such that the data sheet limits are adhered to for voltage levels and power down ramp rate. In addition, the power up ramp must not be too slow, and filtering should be added to the V_{DD} if supply glitches are possible, to avoid upsetting the RAM registers recalled from EEPROM.

If, after all precautions are taken, there still exists the possibility of RAM upset during or after EEPROM POR, a procedure has been outlined here that can be implemented in microcode for checking and correcting these errors. The resulting code makes for a robust solution in challenging power supply environments.

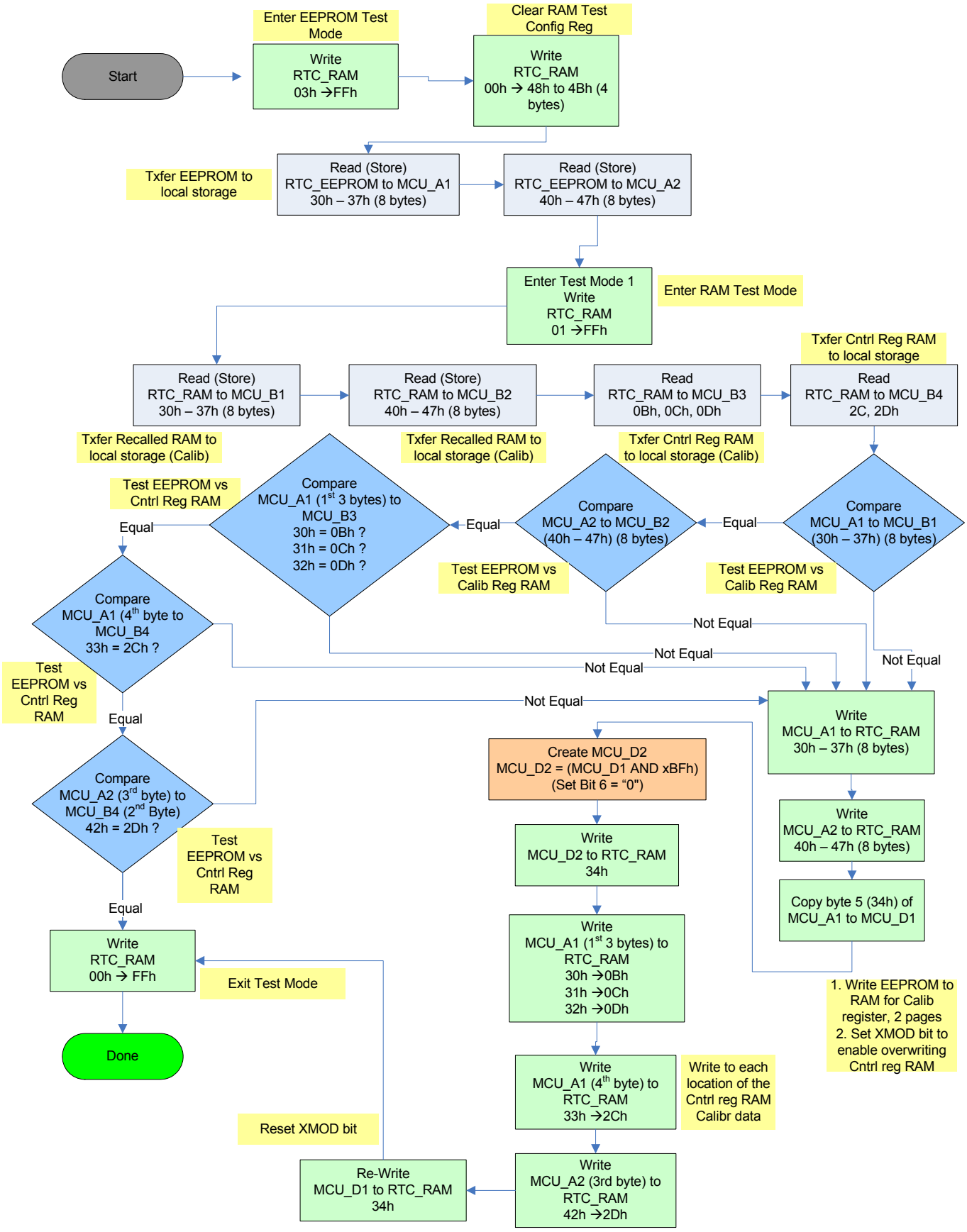

FIGURE 1. EEPROM RECALL FLOW CHART

Appendix

Table 2 is a register map of the ISL12022M. The Different types of registers are described, including RAM, EEPROM/RAM and trim/test/calibration. The yellow highlighted registers are recalled from EEPROM. The light green highlighted registers are duplicates of recalled registers. This is provided for reference only, many of the test and trim registers should not be altered by the user, other than to restore them during the EEPROM POR sequence.

FIGURE 2. ISL12022M REGISTER SUMMARY

ADDR	SECTION	NAME	FUNCTION
00h-06h	RTC		RTC Clock/Calendar Registers
07h	Status	SR	Status Register
08h	Control	INT	Interrupt
09h	Control	PWRVDD	Power Control
0Ah	Control	PWRBAT	Power Control
0Bh	Control	ITR0	Calibration Values from EEPROM, Read-Only
0Ch	Control	ALPHA	
0Dh	Control	BETA	
0Eh	Control	FATR	Calculated Calibration Correction Values, Read-Only
0Fh	Control	FDTR	
10h-15h	Alarm		Alarm Registers
16h-1Fh	TSV2B		Timestamp Registers
20h-27h	DSTCR		Daylight Savings Registers
28h-29h	TEMP		Temperature Registers, Read-Only
2Ah-2Bh	NPPM		NPPM Error Calculation Registers, Read-Only
2Ch	XT0		Calibration Values from EEPROM, Read-Only
2Dh	ALPHA_H		Calibration Values from EEPROM, Read-Only
30h	TRIMR1	IATR/DTR	Initial ATR and DTR Calibration Settings. EEPROM and RAM
31h		TALPHA	Alpha Cold Setting, EEPROM and RAM
32h		TBETA	Beta Settings, EEPROM and RAM
33h		T0	Turnover Temperature Settings, EEPROM and RAM
34h		TsBG	Trim and User Overwrite Enable, EEPROM and RAM
35h		TrBG	Trim, EEPROM and RAM
36h		TPTAT	Trim, EEPROM and RAM
37h		TADC_OS	Trim, EEPROM and RAM
40h	TRIMR2	TCADJP	Trim, EEPROM and RAM
41h		TCADJN	Trim, EEPROM and RAM
42h		ALPHA_H	Alpha hot setting, EEPROM and RAM
43h		Tr_Extra2	Reserved trim, EEPROM and RAM
44h		Tr_Extra3	Reserved trim, EEPROM and RAM
45h		TXT1	Reserved trim, EEPROM and RAM
46h		TXT2	Reserved trim, EEPROM and RAM
47h		TXT3	Reserved trim, EEPROM and RAM
48h	TEST	TMCR0	Test Mode Control, RAM
49h		TMCR1	Test Mode Control, RAM
4Ah		TMCR2	Test Mode Control, RAM
4Bh		TMCR3	Test Mode Control, RAM
FFh	TMEL	TMEL	Test Mode Enable, RAM