Software Implementation Guidelines

Description

This document describes the startup sequence to configure and utilize the fuel gauge functions of MAX77972. The device should be initialized and loaded with a customized model and parameters at power-up. Then the Reported State of Charge and other useful information can be easily read by the host system over I^2C and displayed to the user.

Initialize Registers to Recommended Configuration

The MAX77972 should be initialized before being used. The registers described in this guide should be written to the correct values for the device to perform at its best. These values are written to RAM, so they must be written to the device whenever power is applied or restored. Some registers are updated internally, so verifying that the register was written correctly is necessary to prevent data collisions.

Caution on I^2C Word Size

FG_FUNC and FG_DEBUG are slave addresses with 16-bit word sizes so writing a transaction attempting to modify only 8 bits shall not be attempted since the I^2C control logic discards the transaction.

Step 0 Pull CHGEN Pin Low

It is recommended to start the initialization process with minimum battery current. After the system power-on reset, use one GPIO from the system application processor to pull the CHGEN pin low to turn the charger off.

```
GPIO_CHGEN = Low;//Set CHGEN low to disable charging
```

Step 1 Delay until FSTAT.DNR bit == 0

After power-up, wait for the MAX77972 to complete its startup operations.

```
Wait(10);//wait 10ms
while(ReadRegister(FSTAT_add)&1) Wait(100);
//Wait Loop at 100ms each iteration. Do not continue until FSTAT.DNR==0.
//Typical DNR clear time from POR is 560ms.
```

Step 2 Check for POR and Battery Over Discharge

The POR bit is bit 1 of the Status register and the BAT_dis_OC bit is bit 7 of the ChgDetails01 register.

```
StatusPOR = ReadRegister(Status_add) & 0x0002;
BAT_dis_OC = (ReadRegister(ChgDetails01_add) & 0x0080) >> 7;
if ((StatusPOR=0)&&(BAT_dis_OC=0))
 {goto Step 5.3;}//then go to Step 5.3.
else
 {goto Step 3;}//then do Step 3.
```

Step 3 Dead Battery Check

If the MAX77972 is powered with a dead battery, it is recommended to charge until the battery voltage reaches 2.5V to start initialization.

```
CHGIN_OK = (ReadRegister(ChgDetails00_add) & 0x4000) >> 14;

if (CHGIN_OK){ //Adapter is connected, battery voltage low
 //If battery voltage is lower than prequal threshold
 GPIO_CHGEN = High; //Set CHGEN high to enable charging
 VCell = ReadRegister(VCell_add);
 While (VCell<0x7D00) {
  Wait(20); //wait 20ms if VBAT < 2.5V
  VCell = ReadRegister(VCell_add);
  //If VBAT cannot reach 2.5V for more than 30min
  //Charging is suspended by prequal timeout
 }
 //Set CHGEN pin low to disable charging once battery voltage reaches 2.5V
 GPIO_CHGEN = Low;
}
Goto Step 4; //goto Step 4
```

Step 4 Initialize Configuration

Any battery is supported by one of three types of configuration data. According to the configuration data, only one of the following sections *Step 4.1*, *Step 4.2*, or *Step 4.3* needs execution.

```
nHibCfg = ReadRegister(nHibCfg_add) ; //Store original nHibCfg value
WriteRegister (nHibCfg_add, 0x0) ; // Exit Hibernate Mode
```

Step 4.1 OPTION 1 EZ Config (No INI file is needed)

```
WriteRegister (DesignCap_add , DesignCap) ;  // Write DesignCap
WriteRegister (IChgTerm_add, IChgTerm)  ;  // Write IChgTerm
WriteRegister (VEmpty_add, VEmpty)    ;  // Write VEmpty
WriteRegister (RepCap_add , 0) ;// Write Repcap=0 to prevent FullCapRep changes
//if the model being loaded is lower capacity than the default model

if (ChargeVoltage>4.275)
 WriteRegister (ModelCfg_add, 0x8400) ;     // Write ModelCfg
else
 WriteRegister (ModelCfg_add, 0x8000) ;     // Write ModelCfg

//Poll ModelCfg.Refresh(highest bit),
//proceed to Step 5 when ModelCfg.Refresh=0.
while (ReadRegister(ModelCfg_add)&0x8000) Wait(10) ;
//do not continue until ModelCfg.Refresh==0
```

Proceed to *Step 5*.

Step 4.2 OPTION 2 Custom Short INI without OCV Table

```
WriteRegister (DesignCap_add, DesignCap) ;  // Write DesignCap
WriteRegister (IChgTerm_add, IChgTerm)  ; // Write IChgTerm
WriteRegister (VEmpty_add, VEmpty)    ;  // Write VEmpty
WriteRegister (RepCap_add, 0) ;  // Write Repcap=0 to prevent FullCapRep
changes if the model being loaded is lower capacity than the default model
WriteAndVerifyRegister (LearnCfg_add, LearnCfg)  ;// (Optional in the INI)

WriteRegister (ModelCfg_add, ModelCfg) ;     // Write ModelCfg

//Poll ModelCfg.Refresh(highest bit)
//until it becomes 0 to confirm IC completes model loading
while (ReadRegister(ModelCfg_add)&0x8000) Wait(10) ;
//do not continue until ModelCfg.Refresh==0

WriteRegister (nRCOMP0_add, nRCOMP0)     ; // Write nRCOMP0
WriteRegister (nTempCo_add, nTempCo)     ; // Write nTempCo
WriteRegister (QRTable00_add, QRTable00)  ; // Write QRTable00
WriteRegister (QRTable10_add, QRTable10)  ; // Write QRTable10
WriteRegister (QRTable20_add, QRTable20)  ; // Write QRTable20
WriteRegister (QRTable30_add, QRTable30)  ; // Write QRTable30
```

Proceed to *Step 5*.

Step 4.3 OPTION 3 Custom Full INI with OCV Table

Step 4.3.1 Write/Read/Verify the Custom Model

Once the model is unlocked, the host software must write the 32-bit word model to the MAX77972. The model is located between memory locations 0x80h and 0x9Fh.

```
WriteRegister (RepCap_add, 0) ;  // Write Repcap=0 to prevent some FullCapRep
//changes if the model being loaded is lower capacity than the default model
```

//Actual bytes to transmit will be provided by ADI after cell characterization.

//See *INI* file at the end of this document for an example of the data to be written.

```
Write16Registers (OCVTable0_add, Table [0]) ;
Write16Registers (XTable0_add, Table [1]) ;
```

The model can be read directly back from the MAX77972. Read the 32 words of the model back from the device to verify if it was written correctly. If any of the values do not match, return to *Step 4.3.1*.

```
Read16Registers (OCVTable0_add) ;
Read16Registers (XTable0_add) ;
```

Step 4.3.2 Write Custom Parameters

```
WriteRegister (DesignCap_add, DesignCap)  ; // Write DesignCap

if (saved parameters history exists) {
 LoadFullCapRep = Saved_FullCapRep;
 LoadFullCapNom = Saved_FullCapNom;
 LoadRCOMP0 = Saved_nRCOMP0;
 LoadTempCo = Saved_nTempCo;
 LoadCycles = Saved_Cycles;
}
else {
 LoadFullCapRep = DesignCap;
 LoadFullCapNom = DesignCap;
 LoadRCOMP0 = nRCOMP0;
 LoadTempCo = nTempCo;
 LoadCycles = 0;
}

WriteRegister (FullCapRep_add, LoadFullCapRep)  ; // Write FullCapRep

int Attempt = 0;
do {
 WriteRegister (dPAcc_Add, 0x0C80); // Write dPAcc
 Wait(2); //Wait 2 ms
}
While (((ReadRegister (dPAcc_add) ! = 0xC80)) && attempt++<3) ;

WriteRegister (LearnCfg, LearnCfg_read & 0xFFFD); //Set LearnCfg.MixEn = 0 to
//block FullCapNom updating from MAX77972
WriteRegister (FullCapNom_add, LoadFullCapNom); // Write FullCapNom
WriteRegister (IChgTerm_add, IChgTerm)   ; // Write IchgTerm
WriteRegister (VEmpty_add, VEmpty)     ; // Write VEmpty
WriteRegister (nRCOMP0_add, LoadRCOMP0)     ; // Write RCOMP0
WriteRegister (nTempCo_add, LoadTempCo); // Write TempCo
WriteRegister (Cycles_add, LoadCycles); // Write Cycles
WriteRegister (QRTable00_add, QRTable00)  ; // Write QRTable00
WriteRegister (QRTable10_add, QRTable10)  ; // Write QRTable10
WriteRegister (QRTable20_add, QRTable20)  ; // Write QRTable20
WriteRegister (QRTable30_add, QRTable30)  ; // Write QRTable30
WriteAndVerifyRegister (LearnCfg_add, LearnCfg & 0xFFFD);   // Write LearnCfg
//and keep LearnCfg.MixEn = 0 to block FullCapNom updating from MAX77972
```

Updating Optional Registers

Some or all of the following registers may be optional and may not be included in the INI, this list is provided as an example only. See the provided *INI* file.

```
WriteRegister (nRelaxCfg_add, nRelaxCfg); //Write nRelaxCfg
WriteRegister (Config_add, Config); //Write Config
WriteRegister (MiscCfg_add, MiscCfg); //Write MiscCfg
```

Step 4.3.3 Initiate Model Loading

Set Config2.LdMdl bit to initiate model loading.

```
// Setting the LdMdl bit in the Config2 register
WriteRegister(Config2_add, Config2_read | 0x8000) ;
```

Step 4.3.4 Poll Config2.LdMdl

Poll the LdMdl bit in the Config2 register and proceed to *Step 4.3.5* when the LdMdl bit becomes 0.

```
//Poll Config2.LdMdl(0x0002)
//until it becomes 0 to confirm IC completes model loading
while (ReadRegister(Config2_add) & 0x8000) Wait(10) ;
//do not continue until Config2.LdMdl==0

// Set Config2.LdMdl will clear the content of Config2
// Write Config2 to desired setting after model loading
WriteRegister (Config2_add, Config2); //Write Config2 (optional)
```

Step 4.3.5 Restore nHibCfg and LearnCfg

```
WriteRegister (nHibCfg_add, nHibCfg) ; //Restore nHibCfg
// Restore LearnCfg.MixEn
WriteRegister (LearnCfg_add, LearnCfg_read | 0x0002);
//Proceed to Step 5
```

Step 5 Initialization Complete

Step 5.1 Clear POR bit and Battery Over Discharge Fault

Clear the POR bit and BAT_dis_OC to indicate that the custom model and parameters were successfully loaded.

```
WriteAndVerifyRegister (Status_add, Status_read & 0xFFFD) ;
//Status.POR bit Cleared
WriteAndVerifyRegister (ChgDetails01, ChgDetails01_read & 0xFF7F) ;
//Battery over discharge fault bit Cleared
```

Step 5.2 Set CGTempCo for Internal/External Current Sense

MAX77972 supports both internal and external battery current sense. The default is internal battery current sense. Only one of the following sections in *Step 5.2.1* or *Step 5.2.2* need execution.

Step 5.2.1 OPTION 1: Internal Current Sense

```
WriteRegister (CGTempCo_add, 0x0022) ;
//Set CGTempCo for temperature compensation
```

Step 5.2.2 OPTION 2: External Current Sense

```
WriteRegister (nADCCfg_add, nADCCfg_read | 0x4); //Enable external current sense
WriteRegister (CGTempCo_add, 0x0000); //Set CGTempCo for temperature compensation
```

Step 5.3 Enable Charging

After successfully writing INI in FG locations and clearing the POR bit, make sure to enable charging by pulling the CHGEN pin high.

```
GPIO_CHGEN = High; //Set CHGEN pin high to enable charging
```

Step 5.4 Lock Command (Optional)

After successfully writing INI in FG locations and clearing the POR bit, set the lock bit to prevent maloperation.

```
WriteRegister (USR_add ,0x0001);
WriteRegister (USR_add ,0x0001); // Write 0x0001 twice to lock FG registers
```

Step 5.5 Identify Battery

If the host recognizes the battery pack as one with a saved history, go to *Step 5.7* to restore all the saved parameters, otherwise, continue to *Step 5.6*.

Monitor the Battery

Once the MAX77972 is initialized and customized, the host can read the desired information from the MAX77972 and display that information to the user.

Read the Fuel-Gauge Results

Step 5.6 Read the Reported Capacity and State of Charge (SOC)

The MAX77972 automatically calculates and reports the state of charge of the cell in terms of a percentage and the mAHrs remaining. The Reported State of Charge (RepSOC), as a percent, is read from memory location 0x07 and the Reported Capacity (RepCap), in mAHrs, is read from memory location 0x06.

```
RepCap = ReadRegister(RepCap_add) ;         //Read RepCap
RepSOC = ReadRegister(RepSOC_add) ;         //Read RepSOC
```

The RepSOC_HiByte has a unit of 1%, so the RepSOC_HiByte can be directly displayed to the user for 1% resolution.

Step 5.7 Save Learned Parameters

It is recommended to save the learned capacity parameters every time bit 2 of the Cycles register toggles (so that it is saved every 64% change in the battery) so that if power is lost the values can easily be restored.

```
Saved_RCOMP0 = ReadRegister(nRCOMP0_add);     //Read RCOMP0
Saved_TempCo = ReadRegister(nTempCo_add);     //Read TempCo
Saved_FullCapRep = ReadRegister(FullCapRep_add);     //Read FullCapRep
Saved_Cycles = ReadRegister(Cycles_add);     //Read Cycles
Saved_FullCapNom = ReadRegister(FullCapNom_add);     //Read FullCapNom
```

Step 5.8 Change nFilterCfg.MIX after LearnStage 0x7

It is recommended to monitor LearCfg.LearnStage, once it reaches 0x7, increase nFilterCfg.MIX to increase the voltage gauge mix rate. Make sure to unlock if the fuel gauge is locked in *Step 5.4*.

```
WriteRegister (USR_add ,0x0000) ; //
WriteRegister (USR_add ,0x0000) ; //Optional, if FG is locked
LearnStage = (ReadRegister(LearCfg_add)&0x0070)>>4;
if(LearnStage == 0x7)
{
 WriteRegister(nFilterCfg_add, (nFilterCfg_read&0xF87f)|0x0600);
 //Write nFilterCfg.MIX to 0xC
```

```
}
WriteRegister (USR_add ,0x0001) ; // Optional
WriteRegister (USR_add ,0x0001) ; // Write 0x0001 twice to lock FG registers
```

MAX77972 INI File Format

Option 2 Short Format Example

```
Device=MAX77972
Title=C:/xxxx/1234_1_111111.csv
ModelVersion=8745  //This keeps track of the version of the INI generator

DesignCap=0x1450
ichgterm=0x333
modelcfg=0x8000
QRTable00=0x1050
QRTable10=0x2012
VEmpty=0xa561
RCOMP0=0x004d
TempCo=0x223e
```

Option 3 Long Format Example

```
Device=MAX77972
Title= C:/xxxx/1234_1_111111.csv
ModelVersion=8745

DesignCap=0x06ae
fullsocthr=0x5f05
ichgterm=0x100
modelcfg=0x8410
QRTable00=0x1050
QRTable10=0x0014
QRTable20=0x1300
QRTable30=0x0c00
VEmpty=0x965a
RCOMP0=0x0070
TempCo=0x223e

;;; Begin binary data
;;; This is formatted as 16-bit words, each on a new line.
;;; Numbers are formatted in hex, for example: 0x0000
; Ignore the first 16 words. These are used by EVKit software only
; 16 words. Data starts here for Address 0x80 for use in Step 5
; 16 words. Data starts here for Address 0x90 for use in Step 5
; Ignore the remaining 48 words.
```

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	4/24	Initial release	—