Application Report
SLVA114 – August 2002

Advanced Gas Gauge Host Firmware Flow Chart for the TI Battery Monitor ICs

Doug Williams *Power Management Products*

ABSTRACT

TI advanced battery-monitoring ICs, such as the bq2018, bq2019, and bq2023, are designed to accurately measure the charge and discharge currents in rechargeable battery packs. Intended for pack integration, these devices contain all the necessary functions to form the basis of a comprehensive battery capacity management system in applications such as cellular phones, PDAs, Internet appliances, or other portable products. TI battery monitors work with the host controller in the portable system to implement the battery gas gauging and management system. The host controller is responsible for interpreting the battery monitor data and communicating meaningful battery data to the end-user or power management system.

This document, written around the bq2019, is designed to assist the firmware engineer engaged in development of advanced gas gauging routines in the host controller. A flow chart is presented for implementing the gas gauging function in the host firmware. Suggested constant and variable values are outlined and described.

This document is designed to be used in conjunction with the related application note *Advanced Gas Gauge Host Firmware Guide for the TI Battery Monitor ICs (SLVA100)*

Gas Gauging Functions

Apart from the required general read and write functions, gas gauge firmware may be broken down into six tasks as in table 1. The flowchart is organized according to these functions.

Table 1. Gas Gauge Firmware Tasks

Name	Recommended Interval	Description
GGInitialize( )	On power up	Qualify the battery and communication. Read and convert constants and scratch pad variables from bq2019 into the host.
GGUpdate( )	Once per minute (typical)	Calculate capacity, average current, run time and charge time. Manage full, empty and learning. Update display.
GGRegisterMaint( )	Hourly	Do register maintenance. Store remaining capacity in the bq2019. Clear registers.
GGSelfDischarge( )	Several times per day	Make corrections to remaining capacity for self-discharge.
GGMeasBattVltg( )	Every 20 seconds	Measure the battery voltage. Call GGUpdate( ) if battery crosses the empty voltage threshold.
GGPwrDwnSave( )	On power down	Provide an orderly shutdown. Update bq2019 registers.

Gas Gauging Constants

Table 2 below outlines a set of suggested constants for gas gauge implementation. Notice that the last three are only used as a mechanism to simplify calculations in the firmware. The flowchart refers specifically to many of these constants.

Table 2. Typical Gas Gauge Constants

Class	Name	Suggested bq2019 Address [1]	Used In	Type/Units	Description/Comment
Mfg Data	ID_ROM	78~7F	GGInitialize( )	8 bytes	Factory programmed ROM
	sMFG_DATA	20~25	GGInitialize( )	(str) ASCII	Manufacturers data
	sMODEL	26~2B	GGInitialize( )	(str) ASCII	Battery model
	sMFG_NAME	2C~35	GGInitialize( )	(str) ASCII	Manufacturer name
	iSERIAL_NO	36/37	GGInitialize( )	(uint)	Serial number
	iMFG_DATE	38/39	GGInitialize( )	(uint) Date	Manufactured date
Design Data	iDES_CAP	3A/3B	GGInitialize( ) GGUpdate( )	(uint) Milliampere Hours	Pack design capacity
	iSNS_RES	3C/3D	GGInitialize( )	(uint) mΩ * 327.68	Sense resistor mΩ x 2^{15} / 100
	iSLF_DSG_RATE	3E/3F	GGInitialize( ) GGSelfDischarge	(uint) %/Day * 105.8	Self discharge rate
	iEND_DSG_VLTG	40/41	GGInitialize( ) GGUpdate( ) GGMeasBattVltg( )	(uint) Millivolts	End of discharge voltage
	iTERM_CURR	42/43	GGInitialize( ) GGUpdate( )	(uint) Milliamperes	Charge taper termination current
	iCAP_COMP_TE	44	GGInitialize( ) GGUpdate( )	(byte) % of full capacity	Capacity compensation for temperature
	iCAP_COMP_LD	45	GGInitialize( ) GGUpdate( )	(byte) % of full capacity	Capacity compensation for load current
	iTALK_LD	n/a	GGInitialize( )	(uint) mA	Estimated talk load
	iSTBY_LD	n/a	GGInitialize( )	(uint) mA	Estimated standby load
	iTALK_LD_CNTS	n/a	GGInitialize( ) GGUpdate( )	Counts @ 3.05 µVh	= iSNS_RES * iTALK_LD / 1000
	iSTBY_LD_CNTS	n/a	GGInitialize( ) GGUpdate( )	Counts @ 3.05 µVh	= iSNS_RES * iSTBY_LD / 1000
	iCYCL_CAP	n/a	GGInitialize( ) GGUpdate( )	(uint) = iDES_CAP * iSNS_RES* 0.8/1000	May be used to simplify math in GGUpdate( )

[1] Addresses are in hex. n/a indicates that the values are maintained by the host since they are not a function of the battery. ID_ROM is a fixed address in the bq2019.

Gas Gauging Variables

Table 3, below outlines suggested variables and memory addresses for gas gauge implementation. The flowchart refers specifically to most of these values.

Table 3. Typical Gas Gauge Variables

Class	Name	Suggested bq2019 Address [2][3]	Used In	Type/Units	Description/Comment
Computed Values	iLastMeasDsg	00/01	GGInitialize( ) GGUpdate( ) GGSelfDischarge( )	(uint) Counts @ 3.05 µVh	Last measured discharge. (Initial value set during mfg.)
	iRemCap	02/03	GGInitialize( ) GGUpdate( )	(uint) Counts @ 3.05 µVh	Remaining capacity (Initial value set during mfg.)
	iCycleCnt	04/05	GGUpdate( )	(uint) Units	Cycle count (Typically increased if iDsgCntrCuml has reached 80% of design capacity)
	iMaxTemp	06	GGInitialize( ) GGUpdate( ) GGPwrDwnSave( )	(byte) °K	Max temp seen by this bq2019. Update it in the host during GG_Update.
	iValidDsg	07	GGInitialize( ) GGPwrDwnSave( )	(uint) Flag	Valid discharge flag. This is a flag, but 0x55 is used to indicate true to avoid possible corruption on power down/up cycles.
	iDsgCntr	08/09	GGInitialize( ) GGUpdate( ) GGSelfDischarge( ) GGPwrDwnSave( )	(uint) Counts @ 3.05 µVh	Discharge counter for learning a new iLastMeasDsg
	iDsgCntrCuml	0A/0B	GGInitialize( ) GGUpdate( ) GGSelfDischarge( ) GGPwrDwnSave( )	(uint) Counts @ 3.05 µVh	Cumulative discharge counter tracks partial discharges for iCycleCnt update.
	iLastRemCap	0C/0D	GGInitialize( ) GGPwrDwnSave( )	(uint) Counts @ 3.05 µVh	Last computed remaining capacity value
	iCumlCorrectn	0E/0F	GGInitialize( ) GGUpdate( ) GGSelfDischarge( ) GGPwrDwnSave( )	(uint) Counts @ 3.05 µVh	Tracks cumulative self-discharge corrections. Disqualifies learning cycle if it exceeds 10% of iDES_CAP.
	iTimeSinceMaint	n/a	GGInitialize( ) GGUpdate( ) GGRegisterMaint( )	(uint) Minutes	Minutes elapsed since last register maintenance.
	iRunTime	n/a	GGUpdate( )	(uint) Minutes	Estimated remaining run time at present current.

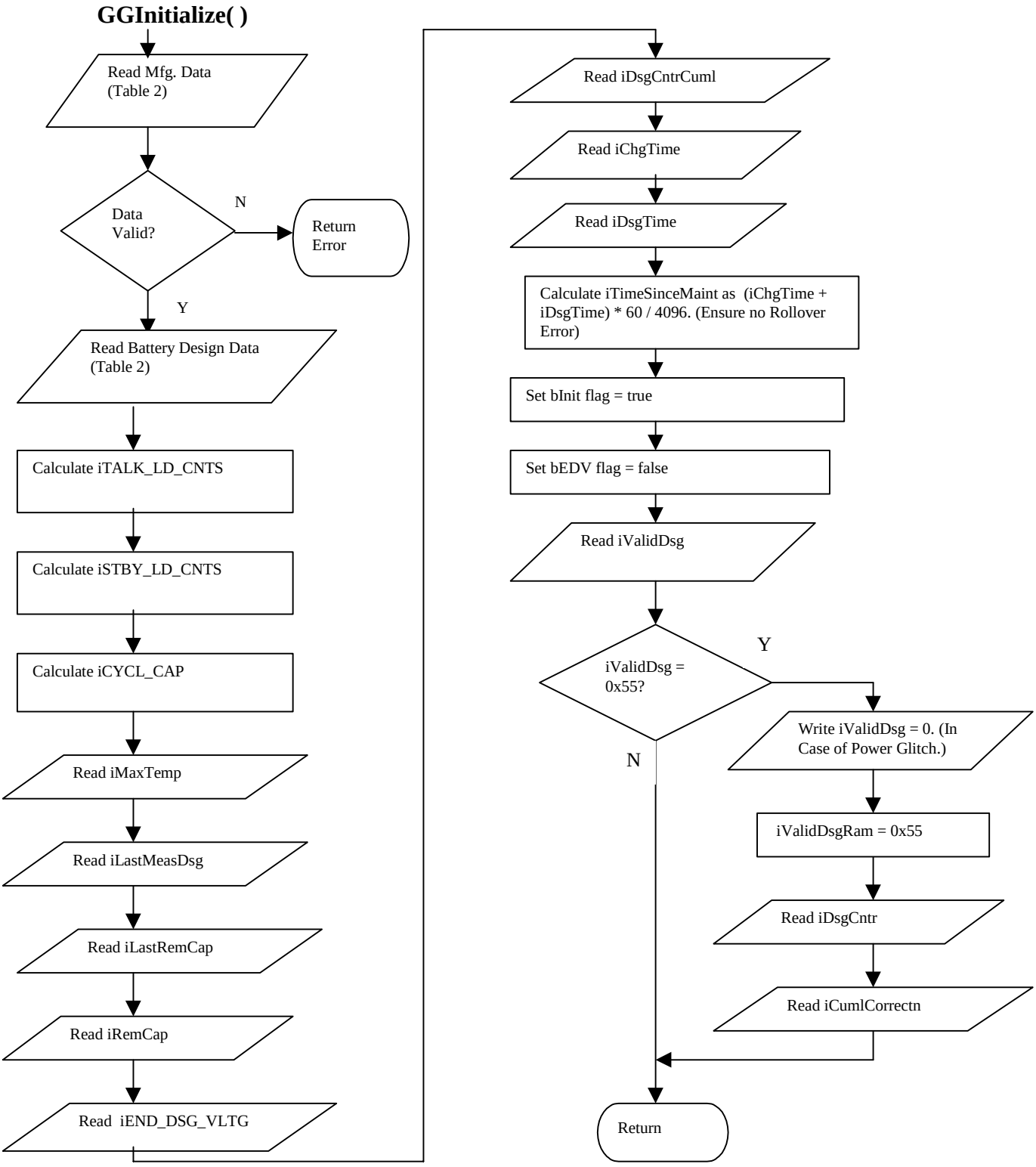
Table 3. Typical Gas Gauge Variables (Continued)

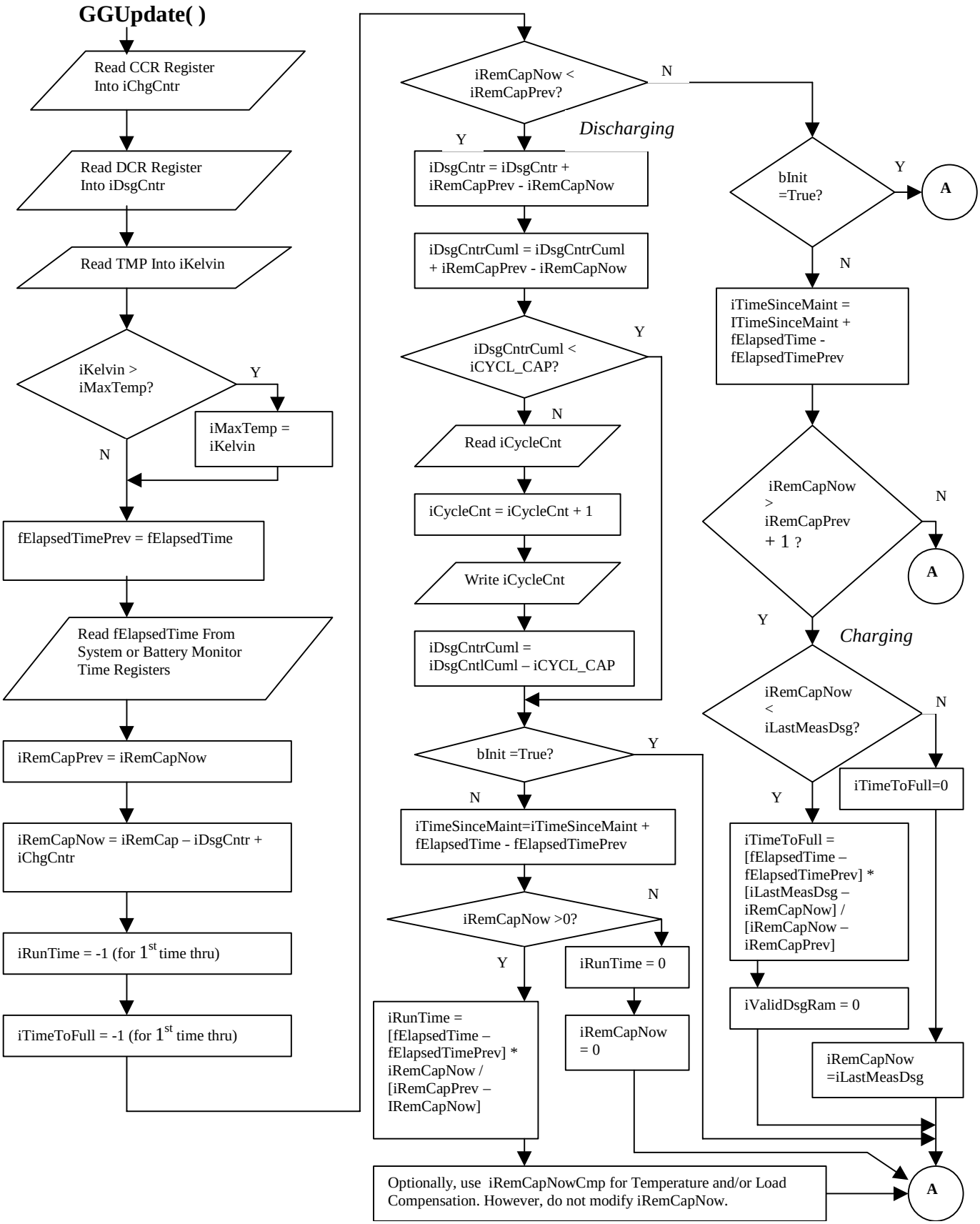
Class	Name	Suggested bq2019 Address [2][3]	Used In	Type/Units	Description/Comment
Computed Values	iTimeToFull	n/a	GGUpdate( )	(uint) Minutes	Estimated remaining time to charge to full.
	iTalkTime	n/a	GGUpdate( )	(uint) Minutes	Estimated remaining run time at iTALK_LD
	iStbyTime	n/a	GGUpdate( )	(uint) Minutes	Estimated remaining run time at iSTBY_LD
	iRelChgPercent	n/a	GGUpdate( )	(uint) %	Relative charge in percent of full charge. aka 'RSOC'
	bInit	n/a	GGInitialize( ) GGUpdate( )	(bool) Flag	Initial pass flag. Inhibits iRunTime and iTimeToFull calculations the first time through GGUpdate( )
	bEDV	n/a	GGInitialize( ) GGUpdate( )	(bool) Flag	Flag to notify other host process that battery is at end of discharge voltage.
	bChgFull	n/a	GGUpdate( )	(bool) Flag	Flag used by the charger or other host process to indicate that the battery is fully charged.
	iValidDsgRam	n/a	GGInitialize( ) GGUpdate( ) GGPwrDwnSave( )	(uint) Flag	Host version of iValidDsg. This is a flag, but 0x55 is used to indicate true to avoid possible corruption on power down/up cycles.
	iRemCapNow	n/a	GGInitialize( ) GGUpdate( ) GGRegisterMaint( ) GGSelfDischarge( ) GGPwrDwnSave( )	(int) Counts @ 3.05 µVh	Most recent calculation of remaining capacity.
	iRemCapNowCmp	n/a	GGUpdate( )	(int) Counts @ 3.05 µVh	If load and/or temperature compensation are used for remaining capacity, this separate variable is required for the most recent calculation of compensated remaining capacity.
	iRemCapPrev	n/a	GGUpdate( )	(int) Counts @ 3.05 µVh	Calculation of remaining capacity from the previous update.
	iRemCapTemp	n/a	GGSelfDischarge( )	(uint) Counts @ 3.05 µVh	Temp variable for iterative self-discharge estimate.
	iSlfDsgEst	n/a	GGSelfDischarge( )	(uint)	Temp self-discharge variable.
	iTempCorrection	n/a	GGSelfDischarge( )	(uint)	Temp self-discharge variable.
	fElapsedTime	n/a	GGUpdate( )	(float) Minutes	Value in minutes, with resolution to seconds since power on reset detected.

Table 3. Typical Gas Gauge Variables (Continued)

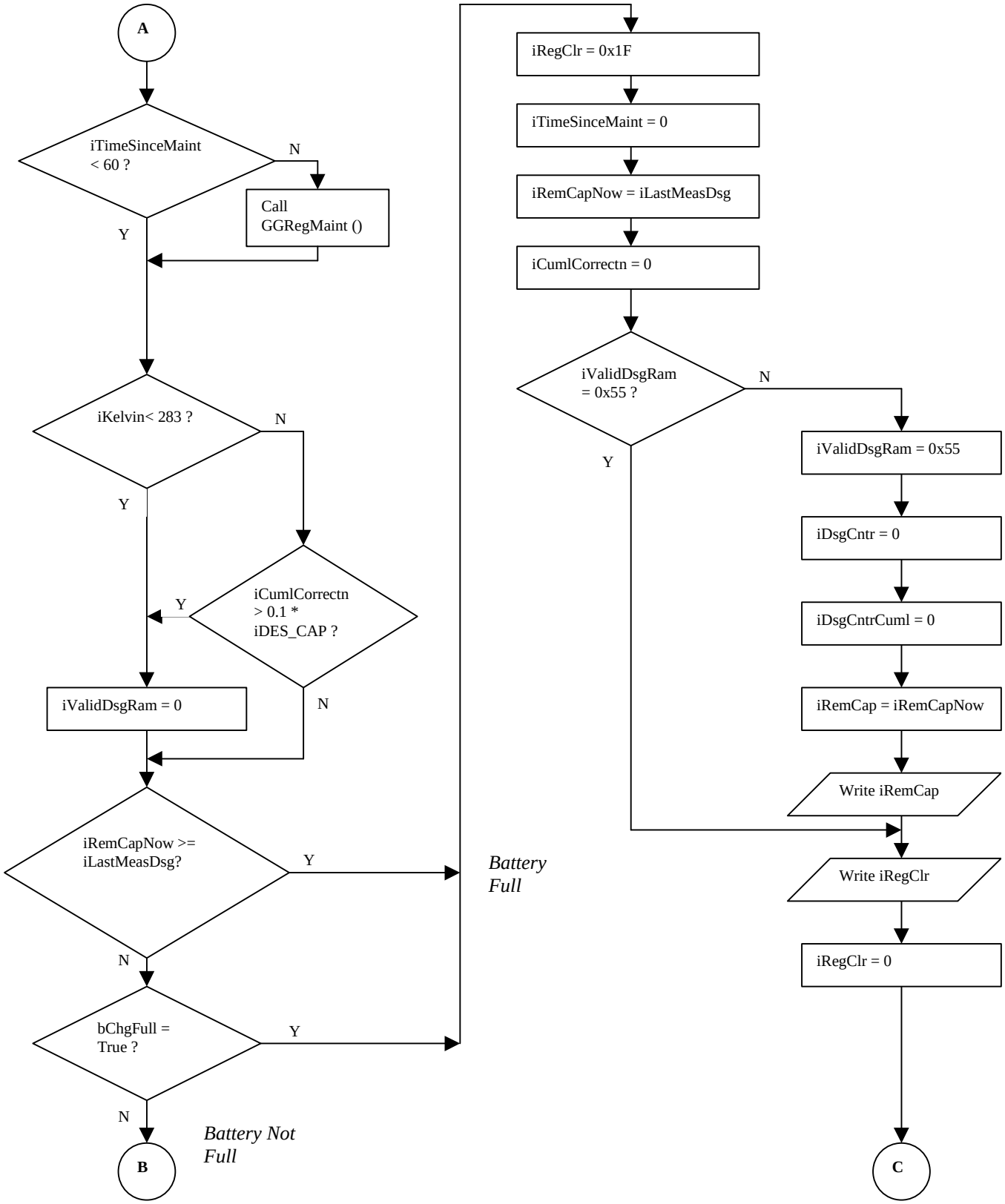
Class	Name	Suggested bq2019 Address [2][3]	Used In	Type/Units	Description/Comment
Computed Values	fElapsedTimePrev	n/a	GGUpdate( )	(float) Minutes	Value in minutes, with resolution to seconds since the power on reset detected previous to fElapsedTime.
bq2019 Gas Gauge Registers and corre-sponding host vars.	iKelvin (Host) TMPL (bq2019) TMPH (bq2019)	 60 61	GGInitialize( ) GGUpdate	(uint) °K	TMPH, TMPL in the bq2019 contain die temperature in °K.
	iRegClr (Host) CLR (bq2019)	 63	GGUpdate( ) GGRegisterMaint( ) GGSelfDischarge( )	(uint) Bits	CLR is used to quickly clear DCR, CCR, SCR, DTC, CTC in any combination. iRegClr is used by the host to set up the next desired clearing pattern.
	iChgTime (Host) CTCL (bq2019) CTCH (bq2019)	 65 66	GGInitialize( ) GGUpdate( ) ??	(uint) 4096 counts per hour	Charge Time Counter. May be used in GGUpdate( ) if no timer or time function is available in the host.
	iDsgTime (Host) DTCL (bq2019) DTCH (bq2019)	 67 68	GGInitialize( ) GGUpdate( ) ??	(uint) 4096 counts per hour	Discharge Time Counter. May be used in GGUpdate( ) if no timer or time function is available in the host
	iSlfDsgCntr (Host) SCRL (bq2019) SCRH (bq2019)	 69 6A	GGInitialize( ) GGSelfDischarge( )	(uint) 1 count per hour (20 – 30°C)	Self Discharge Counter. Rate varies automatically with temperature.
	iChgCntr (Host) CCRL (bq2019) CCRH (bq2019)	 6B 6C	GGUpdate( )	(uint) Counts @ 3.05 µVh	Charge count register in bq2019 increments when voltage at the SR pin is positive.
	iDsgCntr (Host) DCRL (bq2019) DCRH (bq2019)	 6D 6E	GGUpdate( )	(uint) Counts @ 3.05 µVh	Discharge count register in bq2019 increments when voltage at the SR pin is negative.

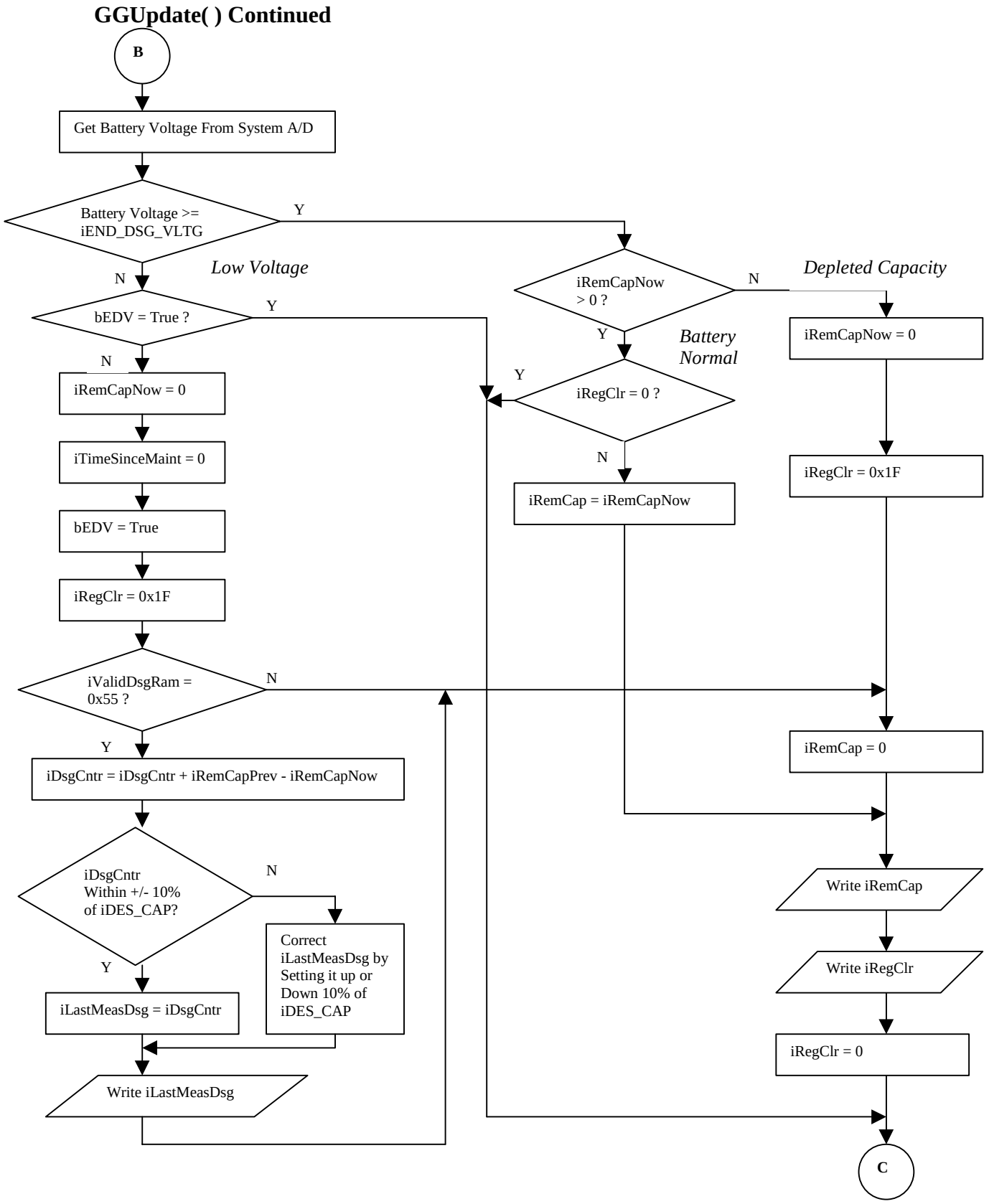
[2] Addresses are in hex. [3]Gas gauge registers are at fixed locations.


GGInitialize( )
Read Mfg. Data (Table 2)
Data Valid?
N
Return Error
Y
Read Battery Design Data (Table 2)
Calculate iTALK_LD_CNTS
Calculate iSTBY_LD_CNTS
Calculate iCYCL_CAP
Read iMaxTemp
Read iLastMeasDsg
Read iLastRemCap
Read iRemCap
Read iEND_DSG_VLTG
Read iDsgCntrCuml
Read iChgTime
Read iDsgTime
Calculate iTimeSinceMaint as (iChgTime + iDsgTime) * 60 / 4096. (Ensure no Rollover Error)
Set bInit flag = true
Set bEDV flag = false
Read iValidDsg
iValidDsg = 0x55?
Y
N
Write iValidDsg = 0. (In Case of Power Glitch.)
iValidDsgRam = 0x55
Read iDsgCntr
Read iCumlCorrectn
Return

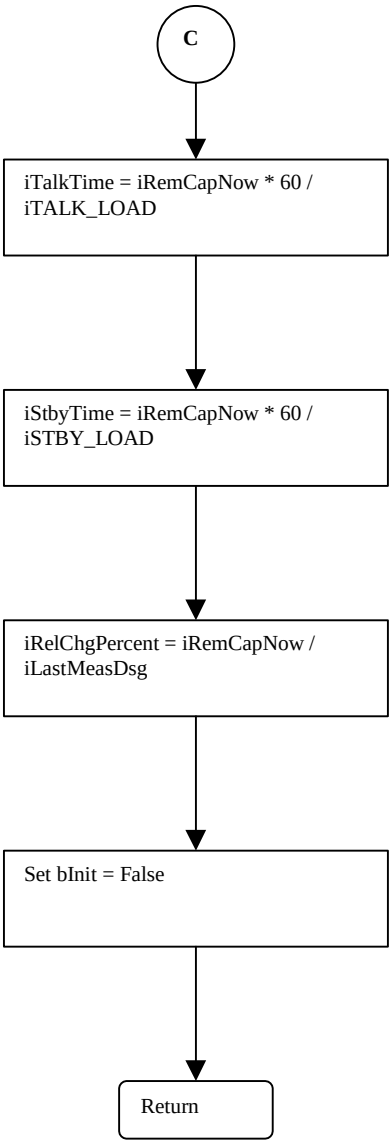

GGUpdate( )
Read CCR Register Into iChgCntr
Read DCR Register Into iDsgCntr
Read TMP Into iKelvin
iKelvin > iMaxTemp?
Y
iMaxTemp = iKelvin
N
fElapsedTimePrev = fElapsedTime
Read fElapsedTime From System or Battery Monitor Time Registers
iRemCapPrev = iRemCapNow
iRemCapNow = iRemCap – iDsgCntr + iChgCntr
iRunTime = -1 (for 1st time thru)
iTimeToFull = -1 (for 1st time thru)
iRemCapNow < iRemCapPrev?
N
Discharging
Y
iDsgCntr = iDsgCntr + iRemCapPrev - iRemCapNow
iDsgCntrCuml = iDsgCntrCuml + iRemCapPrev - iRemCapNow
iDsgCntrCuml < iCYCL_CAP?
Y
N
Read iCycleCnt
iCycleCnt = iCycleCnt + 1
Write iCycleCnt
iDsgCntrCuml = iDsgCntlCuml – iCYCL_CAP
bInit =True?
Y
N
iTimeSinceMaint=iTimeSinceMaint + fElapsedTime - fElapsedTimePrev
iRemCapNow >0?
N
Y
iRunTime = 0
iRemCapNow = 0
iRunTime = [fElapsedTime – fElapsedTimePrev] * iRemCapNow / [iRemCapPrev – IRemCapNow]
bInit =True?
Y
A
N
iTimeSinceMaint = ITimeSinceMaint + fElapsedTime - fElapsedTimePrev
iRemCapNow > iRemCapPrev + 1 ?
N
A
Y
Charging
iRemCapNow < iLastMeasDsg?
N
Y
iTimeToFull=0
iTimeToFull = [fElapsedTime – fElapsedTimePrev] * [iLastMeasDsg – iRemCapNow] / [iRemCapNow – iRemCapPrev]
iValidDsgRam = 0
iRemCapNow =iLastMeasDsg
Optionally, use iRemCapNowCmp for Temperature and/or Load Compensation. However, do not modify iRemCapNow.
A

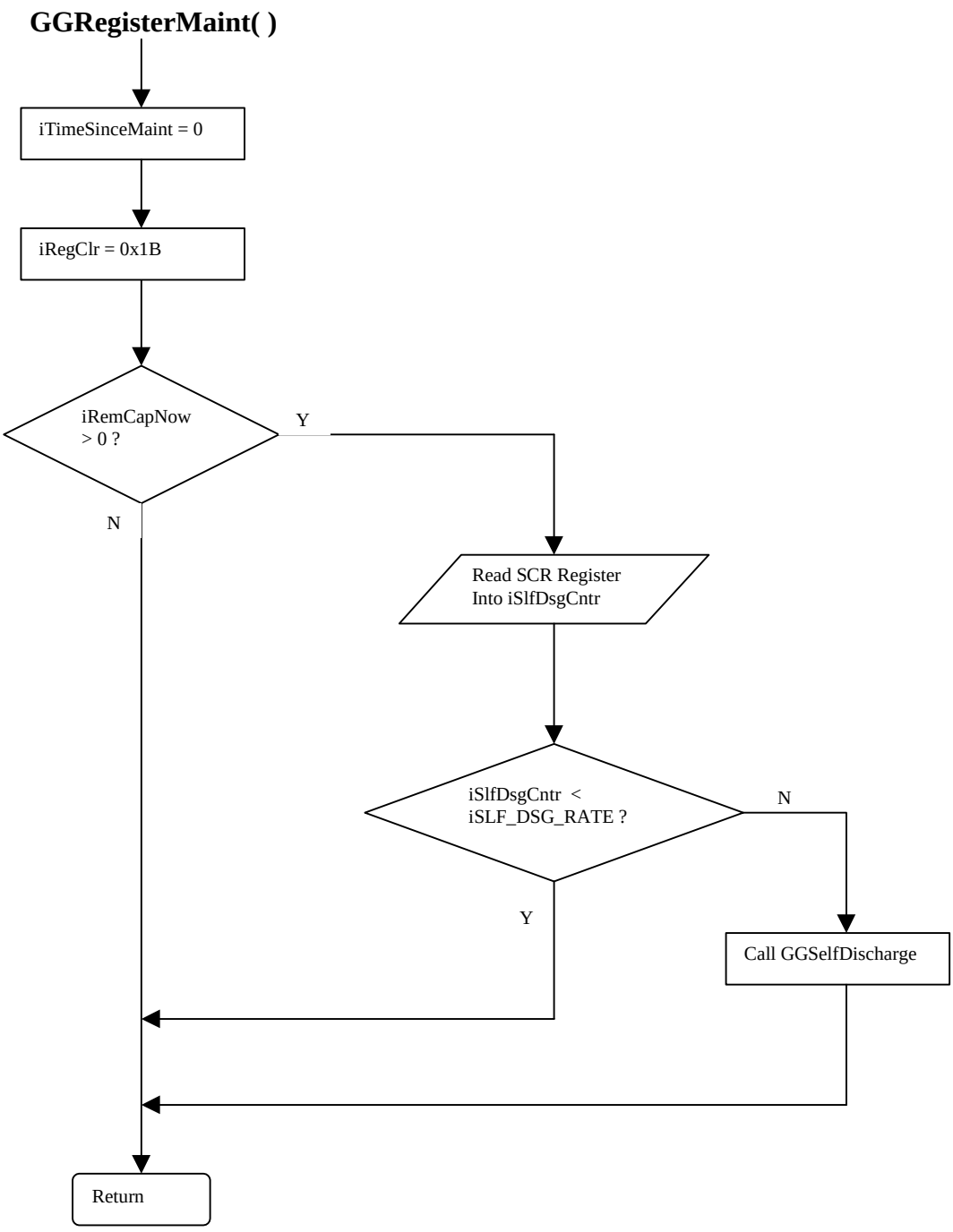
GGUpdate( ) Continued

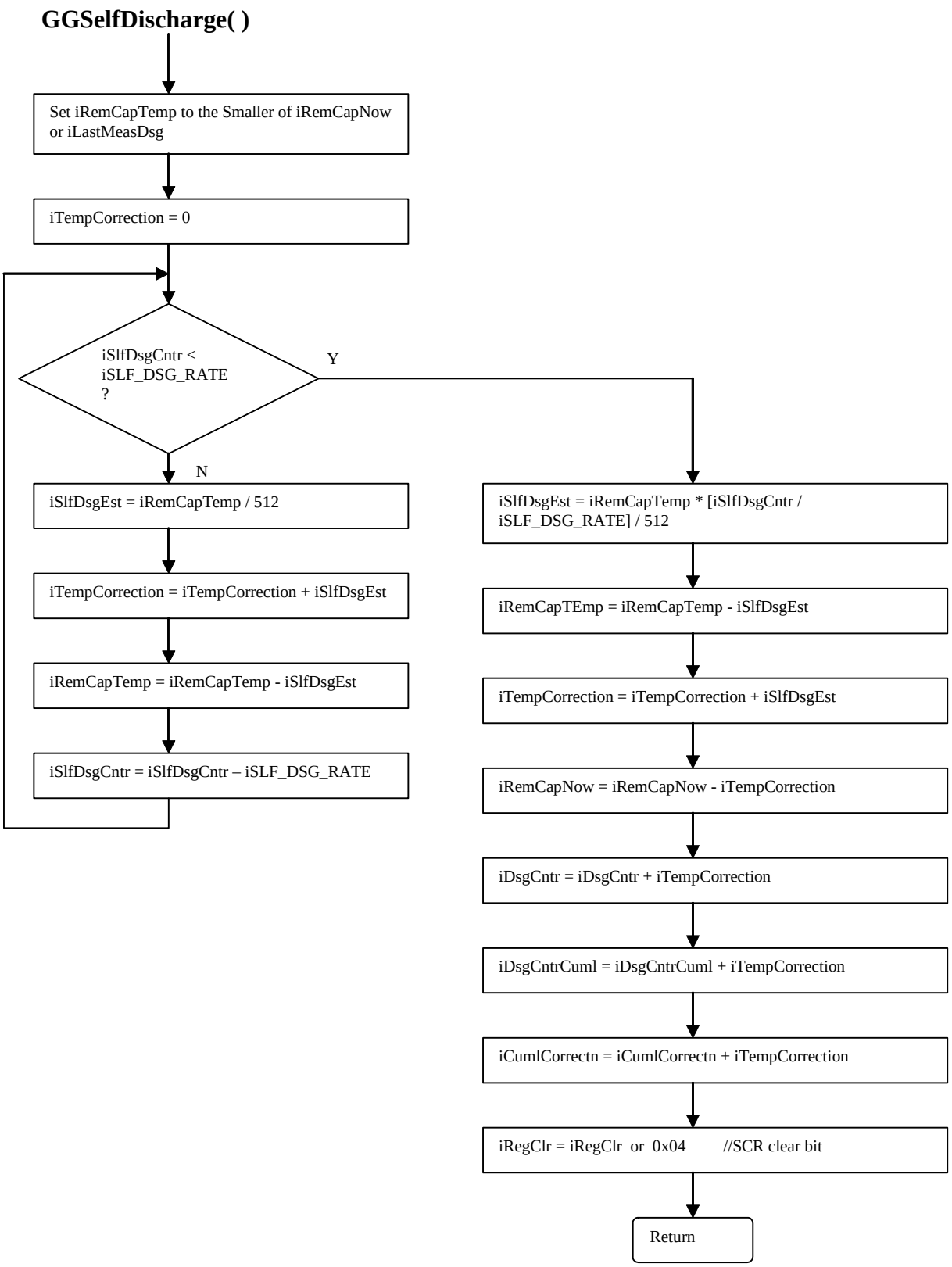


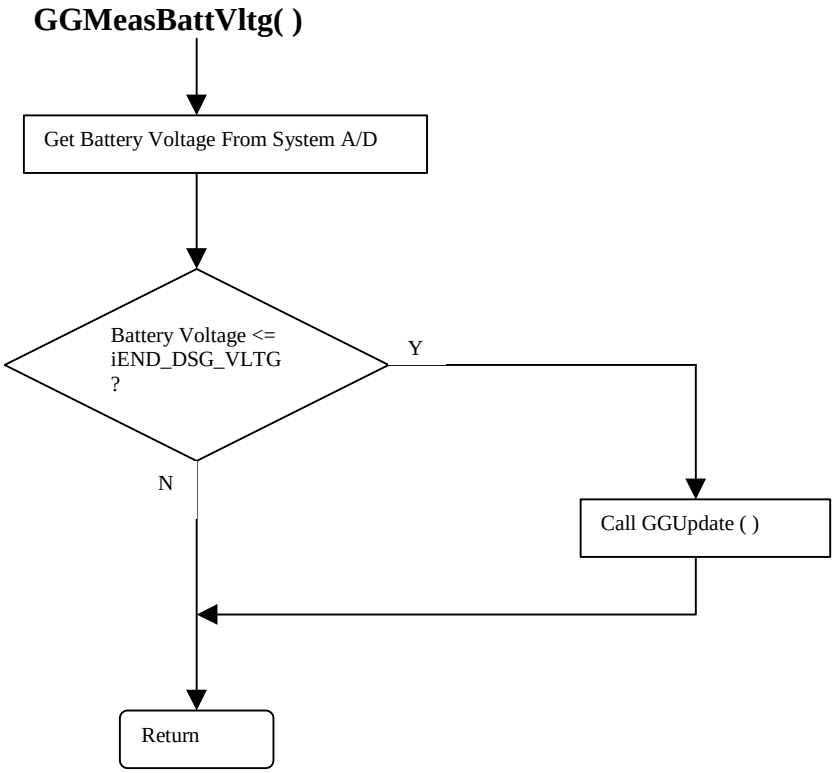

GGUpdate( ) Continued
B
Get Battery Voltage From System A/D
Battery Voltage >= iEND_DSG_VLTG
Y
N
Low Voltage
bEDV = True ?
Y
N
iRemCapNow = 0
iTimeSinceMaint = 0
bEDV = True
iRegClr = 0x1F
iValidDsgRam = 0x55 ?
N
Y
iDsgCntr = iDsgCntr + iRemCapPrev - iRemCapNow
iDsgCntr Within +/- 10% of iDES_CAP?
N
Y
Correct iLastMeasDsg by Setting it up or Down 10% of iDES_CAP
iLastMeasDsg = iDsgCntr
Write iLastMeasDsg
iRemCapNow > 0 ?
N
Depleted Capacity
Y
Battery Normal
iRegClr = 0 ?
Y
N
iRemCap = iRemCapNow
iRemCapNow = 0
iRegClr = 0x1F
iRemCap = 0
Write iRemCap
Write iRegClr
iRegClr = 0
C

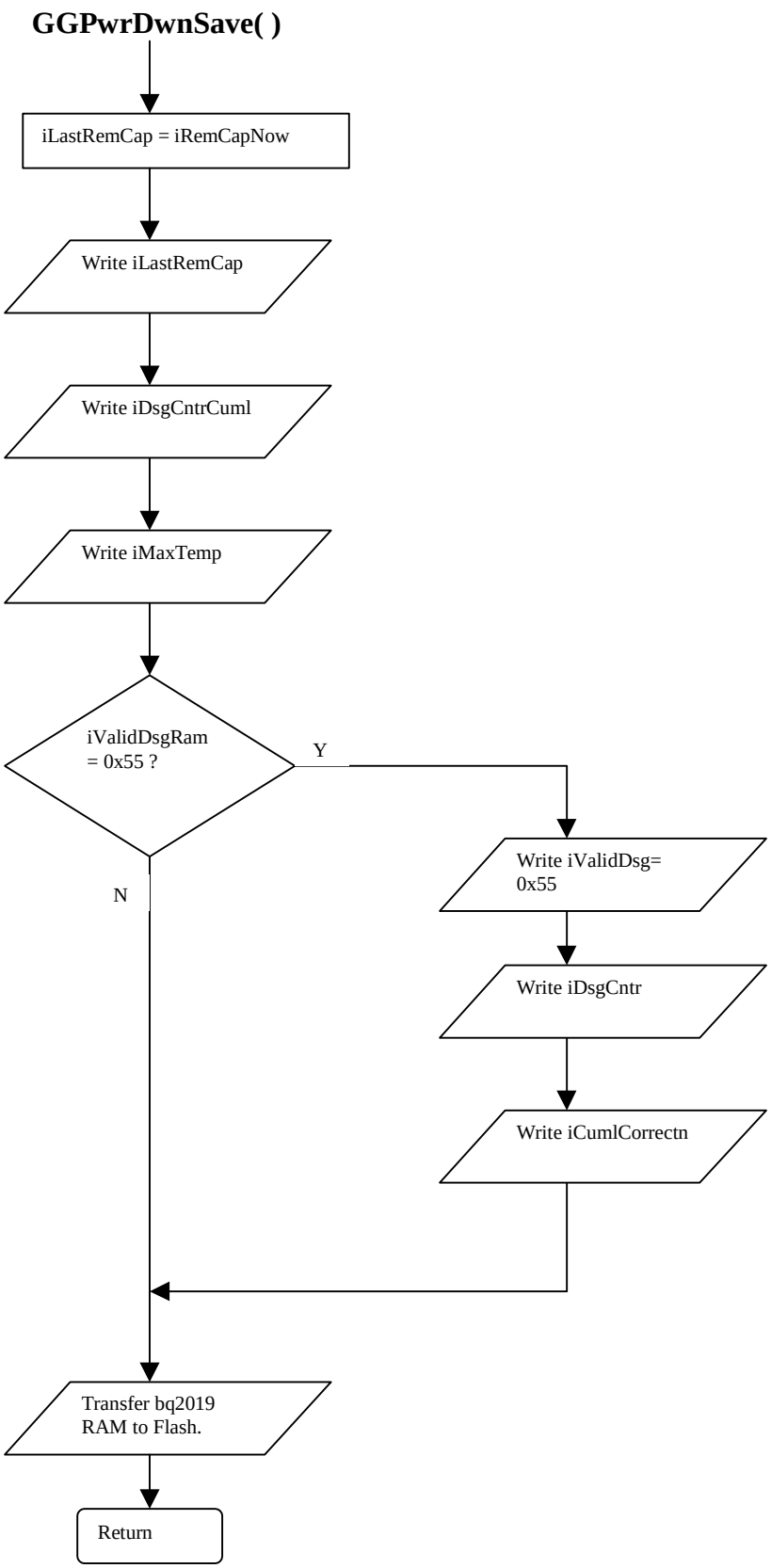
GGUpdate( ) Continued




GGRegisterMaint( )
iTimeSinceMaint = 0
iRegClr = 0x1B
iRemCapNow
> 0 ?
Y
N
Read SCR Register
Into iSlfDsgCntr
iSlfDsgCntr <
iSLF_DSG_RATE ?
N
Y
Call GGSelfDischarge
Return


GGSelfDischarge( )
Set iRemCapTemp to the Smaller of iRemCapNow or iLastMeasDsg
iTempCorrection = 0
iSlfDsgCntr < iSLF_DSG_RATE ?
Y
N
iSlfDsgEst = iRemCapTemp / 512
iTempCorrection = iTempCorrection + iSlfDsgEst
iRemCapTemp = iRemCapTemp - iSlfDsgEst
iSlfDsgCntr = iSlfDsgCntr – iSLF_DSG_RATE
iSlfDsgEst = iRemCapTemp * [iSlfDsgCntr / iSLF_DSG_RATE] / 512
iRemCapTEmp = iRemCapTemp - iSlfDsgEst
iTempCorrection = iTempCorrection + iSlfDsgEst
iRemCapNow = iRemCapNow - iTempCorrection
iDsgCntr = iDsgCntr + iTempCorrection
iDsgCntrCuml = iDsgCntrCuml + iTempCorrection
iCumlCorrectn = iCumlCorrectn + iTempCorrection
iRegClr = iRegClr or 0x04 //SCR clear bit
Return


GGMeasBattVltg( )
Get Battery Voltage From System A/D
Battery Voltage <= iEND_DSG_VLTG ?
Y
N
Call GGUpdate ( )
Return


GGPwrDwnSave( )
iLastRemCap = iRemCapNow
Write iLastRemCap
Write iDsgCntrCuml
Write iMaxTemp
iValidDsgRam
= 0x55 ?
Y
N
Write iValidDsg=
0x55
Write iDsgCntr
Write iCumlCorrectn
Transfer bq2019
RAM to Flash.
Return

References

1. *Texas Instruments. data sheet for bq2019 Advanced Battery Monitor IC (SLUS456)*
2. *Texas Instruments. Advanced Gas Gauge Host Firmware Guide for the TI Battery Monitor ICs (SLVA100)*

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

Copyright © 2002, Texas Instruments Incorporated