19-1843; Rev 0; 12/00

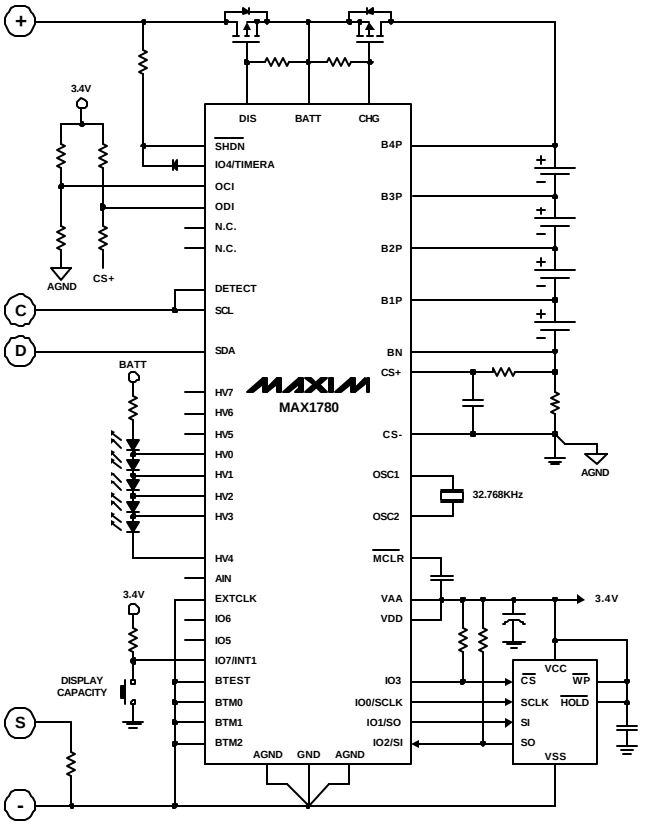
# MAX1780 Advanced Smart Battery Pack Controller

## General Description

The MAX1780 Advanced Battery Pack Controller is a smart battery pack supervisor that integrates a User programmable Microcontroller Core, Coulomb-Counter-based Fuel Gauge, a multi-channel Data Acquisition Unit, a high-speed SPI Interface, and a Master/Slave SMBus Interface. The 8-bit, RISC, Microcontroller core is user programmable, and provides battery pack designers with complete flexibility in developing fuel gauging and control algorithms. The Data Acquisition Unit can measure individual cell voltages to within 50mV, total battery stack voltage (up to 20.48V), and chip internal/external temperature.

The user adjustable overcurrent comparators, along with individual cell voltage measurements, allow the MAX1780 to eliminate a separate primary pack protection IC. The MAX1780 can be directly connected to 2 to 4 series Lithium Ion cells, and supplies itself through a fully integrated 3.4V low drop out linear regulator.

## Typical Operating Circuit




## Features

- User Programmable Using an External EEPROM
- Accurate Fuel Gauge Uses V-to-F Method
  - o < 1µV Input Offset Voltage
  - o No External Calibration Required
- Eliminates Separate Primary Protection IC
  - o 50mV Accurate Individual Cell Voltage Measurements
  - o Built in Protection MOSFET Gate Drivers
  - o Over Charge & Discharge Current Protection
- Fully Integrated LDO ($V_{IN}$ = 4V to 28V)
- 8-bit RISC Microcontroller Core
  - o On-board 1.5K ROM & 0.5K Program RAM
  - o 144 Bytes Data Memory
  - o Fast Start-up 3.5MHz Instruction Oscillator
  - o Watch Dog Timer
- Hardware SMBus with Master Capability
- GPIO Port and High Voltage LED Drivers
- Typical Operating Currents < 200µA Achievable
- Typical Shutdown Current of 1nA

## Applications

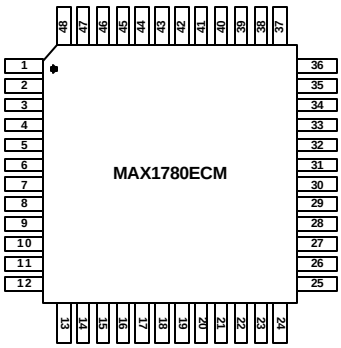
SMBus Battery Packs

Proprietary Battery Packs

Data Acquisition Systems

Remote Data Logging

## Pin Configuration




48-PIN TQFP

## Ordering Information

| PART | TEMP RANGE | PIN/ PACKAGE |
|---|---|---|
| MAX1780ECM | -40 °C to +85 °C | 48 TQFP |

# Advanced Smart Battery Pack Controller

# Table Of Contents

# Table Of Figures

# Table Of Tables

# List Of Applicable Documents

**"System Management Bus Specification"**, Revision 1.1, dated December 11, 1998.

**"Smart Battery Data Specification"**, Revision 1.1, dated December 11, 1998.

**"MAX1780 ROM Code Supplement"**, (Contact the Factory for availability)

**"MAX1780 Programmers Reference Manual"**, (Contact the Factory for availability)

# Architectural Overview

## Introduction

The MAX1780 Advanced Smart Battery Pack Controller was designed to provide Smart Battery pack designers with a flexible solution that accurately measures individual cell characteristics. It combines an 8-bit RISC microprocessor core, program and data memory, with an arsenal of precision analog peripherals. Together with an external serial EEPROM, the MAX1780 provides the most integrated solution for developing battery pack electronics in the industry.

## Harvard Architecture

The MAX1780 uses a Harvard architecture in which program and data are accessed on separate buses. This improves bandwidth over traditional von Neumann architectures where program and data are fetched on the same bus. Separating program and data memory further allows instructions to be sized differently than the 8-bit wide data word. A 12-bit wide program memory bus fetches a 12-bit instruction in a single cycle. The 12-bit wide instruction opcodes make it possible to have all single word instructions.

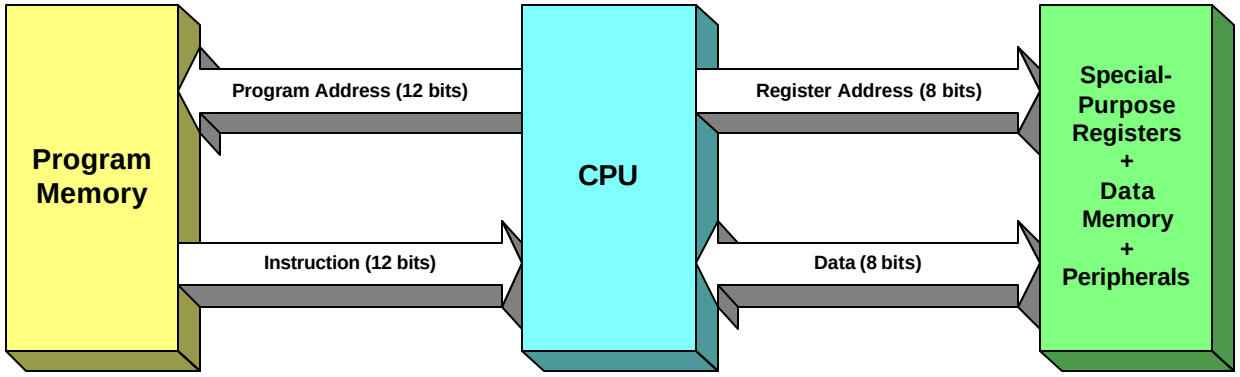



**Figure 1, MAX1780 Harvard Architecture**

The MAX1780 addresses 2K x 12 of internal program memory, organized as three blocks of ROM, each 512 x 12, and one program RAM block 512 x 12. Using special ROM routines (see MAX1780 ROM Supplement for details), the MAX1780 can access up to 64K x 8 of external serial EEPROM memory. This provides software designers with the ability to load program code residing in external EEPROM into program RAM under program control. This innovative capability can be used to develop self-adapting battery management and control software.

Figure 2 shows a block diagram of the MAX1780 core microcontroller. MAX1780 memory is organized into program memory and data memory. Program memory pages are accessed using one or two STATUS register bits. Data memory is composed of RAM, organized as 8, 16 byte groups or pages. Collectively, all of the DRAM pages are called the Register File. The register file is divided into two functional groups: special function registers and general purpose registers. DRAM is accessed using the File Selection Register (FSR).

The special function registers include the TIMERA Register, the Program Counter (PC), the Status Register, the SETUPINT Register, the I/O registers (ports), and the File Select Register (FSR). In addition, special purpose registers are used to control the I/O port configuration and prescaler options. The General Purpose registers are used for data and control information under command of the instructions.

PORTB is used as a dedicated internal interface between the processor core and all the on-board peripherals. Most of the on-board peripherals function autonomously, generating interrupts to the processor core when service is required.

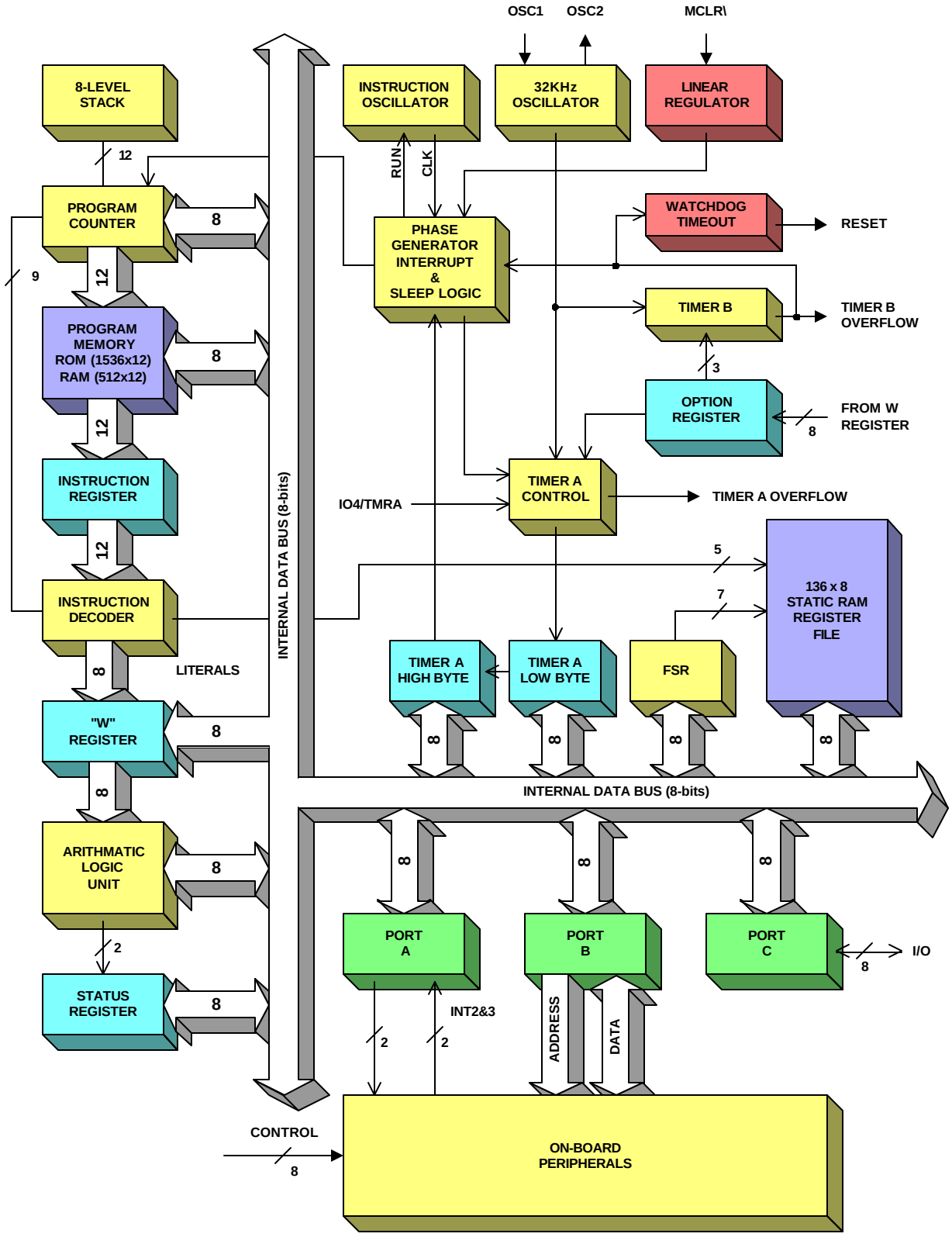



**Figure 2, MAX1780 Block Diagram**

# Detailed Description Of MAX1780

## Instruction Execution

The MAX1780 processor core incorporates a two-stage pipeline. The instruction fetch and execution are pipelined such that an instruction fetch takes one instruction cycle while the instruction decode and execution takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change, for example a GOTO instruction, then two cycles are required to complete the instruction, breaking the pipeline's flow. Figure 3 shows how instruction pipelining increases the processor throughput.

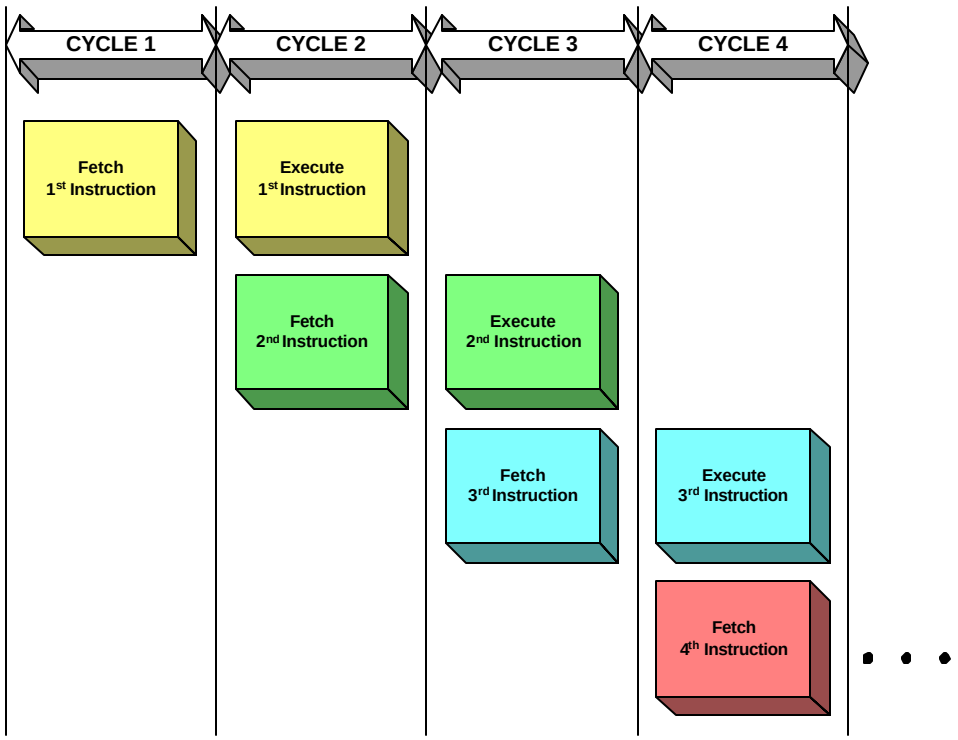



Figure 3, MAX1780 Instruction Pipeline

## Phase Clocks

As shown in Figure 4, an Instruction Cycle consists of four phase clocks. The phase clocks do not overlap, and each phase transition occurs on the rising edge of the Instruction Oscillator. Phase 0 (PH0) is called the Decode Phase, where fetched instructions are decoded. Phase 1 (PH1) is known as the Data Read Phase. Any data from Data Memory required to complete the instruction will be read at this time. Phase 2 (PH2) is called the Process Phase, in which any data retrieved in PH1 is processed. The final phase, Phase 3 (PH3), is the Data Write Phase and is used to write data manipulated during PH2 back to Data Memory.

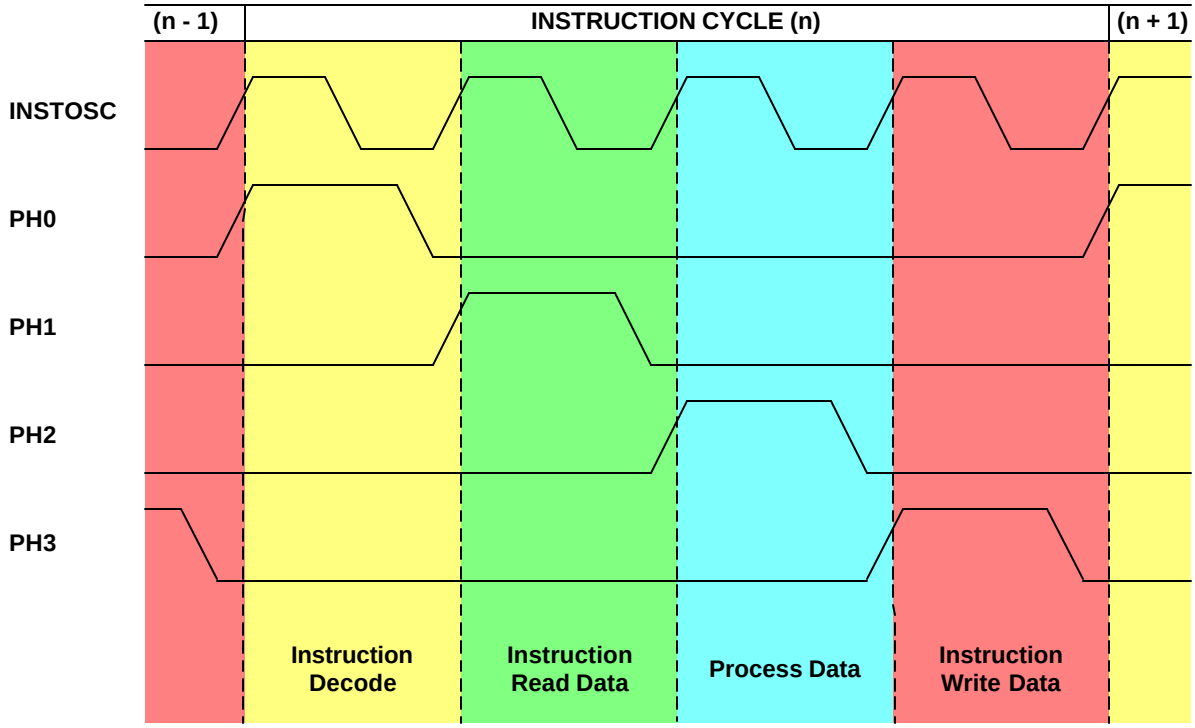



Figure 4, Instruction Cycle Phase Clocks

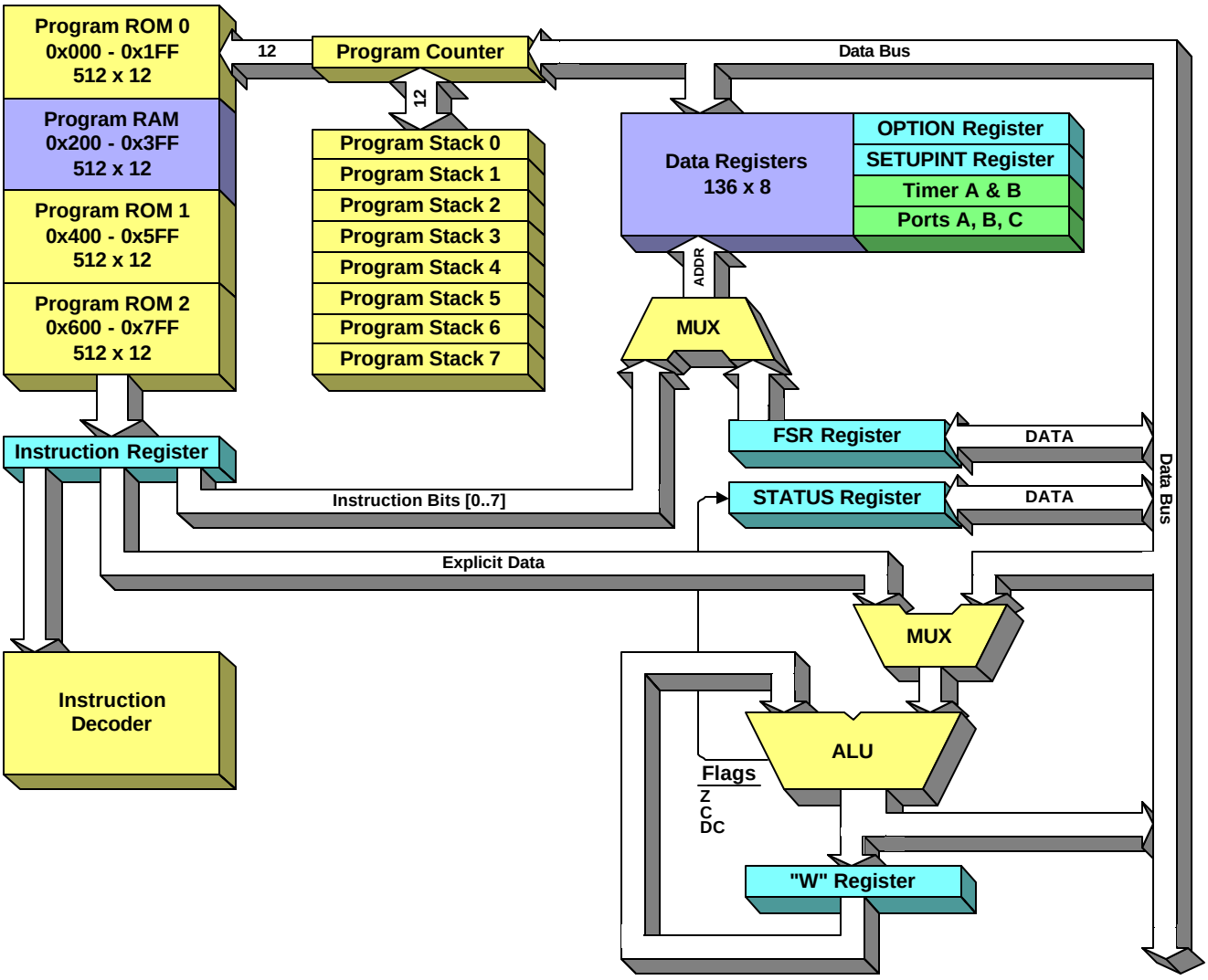



**Figure 5, MAX1780 Programming Model**

# CPU Architecture

**Figure 5** shows the MAX1780's central processing unit (CPU) architecture. As shown, instructions in program memory and variables in data memory are accessed using separate buses

## Program Counter (PC)

As a program instruction is executed, the Program Counter (PC) will contain the address of the next program instruction to be executed. The PC value is increased by one every instruction cycle, unless an instruction changes the PC. It should be noted that all subroutine calls are limited to the first 256 locations of any program memory page (512 words long).

## Program Stack

The MAX1780 has a 12-bit wide, eight-level hardware stack. This allows program developers to create code with nested subroutine calls. When CALL instructions are executed, pushing the current value onto the stack saves the processor context. A word of caution, the MAX1780 program stack has only 8-levels. Programmers should not create code that has unlimited nesting of subroutine calls, or a stack overflow is possible.

A subroutine call is completed with a RETURN or RETLW instruction, both of which will pop the contents of the stack into the program counter. The RETLW instruction also loads the “W” Register with the literal value specified in the instruction. This is particularly useful for the implementation of data look-up tables within the program memory.

## Arithmetic Logic Unit (ALU)

The MAX1780 CPU contains an 8-bit ALU and working register. The ALU is a general-purpose arithmetic unit. It performs arithmetic and Boolean functions between data in the working register and any register file.

The ALU is 8-bits wide and capable of addition, subtraction, shift and logical operations. Unless otherwise mentioned, arithmetic operations are two's complement in nature. In two-operand instructions, typically one operand is the “W” (working) Register. The other operand is either a file register or an immediate constant. In single operand instructions, the operand is either the W Register or a file register.

Depending on the instruction executed, the ALU may affect the values of the Carry (C), Digit Carry (DC), and Zero (Z) bits in the STATUS register.

## Working Register (“W”)

The “W” Register serves as an accumulator or temporary storage register for many instructions. The “W” Register is not directly accessible. Its contents must be moved to other registers that are directly accessible, in order to be read. The “W” Register is also used in every arithmetic operation.

## Option Register (Write Only, via the "W" Register and OPTION Instruction)

The Option Register is a CPU core register. It is not directly accessible and is not in the Register File address space. To change the contents of the Option Register, use the OPTION instruction, which loads the contents of the "W" Register into the Option Register. Through the Option Register the user has access to the Timer A and Timer B controls. Additionally, whether the Instruction Oscillator runs in SLEEP Mode or not may be selected.

Since the contents of the Option Register cannot be read, it is suggested that programmers "shadow" its contents in a global software variable, and all changes be made to the global variable. The global variable is then moved to the "W" Register and an OPTION Instruction executed to affect changes in the Option Register.

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | **OSLB** | **TMRAD** | **TAS1** | **TAS0** | **MSKW** | **PS2** | **PS1** | **PS0** |
| POR STATE | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| **OSLB** | |
|---|---|
| 0 | The Internal Instruction Execution Oscillator Is Turned off During SLEEP Mode. INTOSC OFF in SLEEP. (See note below) |
| 1 | The Internal Instruction Execution Oscillator Runs During SLEEP Mode. INTOSC ON in SLEEP. |

| **TMRAD** | |
|---|---|
| 0 | Timer A (TMRA) Enabled |
| 1 | Timer A (TMRA) Disabled |

| **TAS1** | **TAS0** | |
|---|---|---|
| 0 | 0 | TMRA increments count on the falling edge of IO4/TMRA pin |
| 0 | 1 | TMRA increments count on the rising edge of IO4/TMRA pin |
| 1 | 0 | TMRA increments count on the rising edge of the 32KHz Clock |
| 1 | 1 | TMRA increments count on rising edge of INTOSC/4 |

| **MSKB** | |
|---|---|
| 0 | TIMER B (TMRB) overflow generates an interrupt if INTOFF = 0 (cleared in STATUS Register) |
| 1 | TIMER B (TMRB) overflow does not generate an interrupt |

| **PS2** | **PS1** | **PS0** | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | TMRB period = (1/32KHz) * 256 * 2 | 15.62 | ms |
| 0 | 0 | 1 | TMRB period = (1/32KHz) * 256 * 4 | 31.25 | ms |
| 0 | 1 | 0 | TMRB period = (1/32KHz) * 256 * 8 | 62.50 | ms |
| 0 | 1 | 1 | TMRB period = (1/32KHz) * 256 * 16 | 125.0 | ms |
| 1 | 0 | 0 | TMRB period = (1/32KHz) * 256 * 32 | 250.0 | ms |
| 1 | 0 | 1 | TMRB period = (1/32KHz) * 256 * 64 | 500.0 | ms |
| 1 | 1 | 0 | TMRB period = (1/32KHz) * 256 * 128 | 1.0 | sec |
| 1 | 1 | 1 | *FACTORY USE ONLY (DO NOT USE) | - | - |

Note: The instruction oscillator will continue to run in SLEEP while an SMBus Slave transmission is in progress.

## Setupint Register (Write Only, via the "W" Register and FREE Instruction)

The Setupint Register is a CPU core register. It is not directly accessible and is not in the Register File address space. To change the contents of the Setupint Register, use the FREE instruction with FSR bits 5 and 6 cleared to '0', which loads the contents of the "W" Register into the Setupint Register. The Setupint Register is used to set-up the triggering characteristics of the MAX1780's three interrupt channels. Additionally, Bit 0 is the global enable bit for addressing the on-board peripherals.

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | **SENSE3** | **SENSE2** | **SENSE1** | **EN1SN** | **EDGE3** | **EDGE2** | **EDGE1** | **PADDR** |
| POR STATE | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### Interrupt 3

| EDGE3 | SENSE3 | |
|---|---|---|
| 0 | 0 | IF3 Flag in the PORTA Register is set to '1' when INT3 = '0'. |
| 0 | 1 | IF3 Flag in the PORTA Register is set to '1' when INT3 = '1'. |
| 1 | 0 | IF3 Flag in the PORTA Register is set to '1' on a Falling Edge of INT3. |
| 1 | 1 | IF3 Flag in the PORTA Register is set to '1' on a Rising Edge of INT3. |

Note: The Peripheral Interrupt Controller uses INT3 for interrupts. There are total of 8 INT3 interrupt sources: POWER FAILURE, FGCHGSTAT, ADCDONE, DETECT, OCICMP, ODICMP, FGCHGOFL, FGDISOFL.

### Interrupt 2

| EDGE2 | SENSE2 | |
|---|---|---|
| 0 | 0 | IF2 Flag in the PORTA Register is set to '1' when INT2 = '0'. |
| 0 | 1 | IF2 Flag in the PORTA Register is set to '1' when INT2 = '1'. |
| 1 | 0 | IF2 Flag in the PORTA Register is set to '1' on a Falling Edge of INT2. |
| 1 | 1 | IF2 Flag in the PORTA Register is set to '1' on a Rising Edge of INT2. |

Note: The SMBus Interface uses INT2 for interrupts. There are a total of 7 INT2 interrupt sources: START, RESTART, STOP, SCLHOLD, TOUT, MSTON, ACK.

### Interrupt 1

| EN1SN | EDGE1 | SENSE1 | |
|---|---|---|---|
| 0 | 0 | 0 | Do not clear the EN1SN bit to '0'. |
| 0 | 0 | 1 | Do not clear the EN1SN bit to '0'. |
| 0 | 1 | 0 | Do not clear the EN1SN bit to '0'. |
| 0 | 1 | 1 | Do not clear the EN1SN bit to '0'. |
| 1 | 0 | 0 | IF1 Flag in the PORTA Register is set to '1' when the IO7/INT1 pin = '0'. |
| 1 | 0 | 1 | IF1 Flag in the PORTA Register is set to '1' when the IO7/INT1 pin = '1'. |
| 1 | 1 | 0 | IF1 Flag in the PORTA Register is set to '1' on an IO7/INT1 Falling Edge. |
| 1 | 1 | 1 | IF1 Flag in the PORTA Register is set to '1' on an IO7/INT1 Rising Edge. |

Note: Do not clear the EN1SN bit to '0', otherwise this could result in a constant interrupt from INT1.

### Peripheral Address Bit

| PADDR | |
|---|---|
| D0 | This bit must be set to '1' to enable access to on-board peripherals. |

Note: This bit defaults to '1' on Power-On-Reset. Do not change or the on-board peripherals cannot be addressed.

## Memory Organization:

The MAX1780 program memory is divided into four blocks 512 x 12-bits in size. Each block is further divided into 256 x 12-bits upper and lower pages. The first, third, and fourth blocks are Program ROM, which are programmed at the factory. The second memory block is Program RAM, which is loaded with user code at boot, and can be modified during operation.

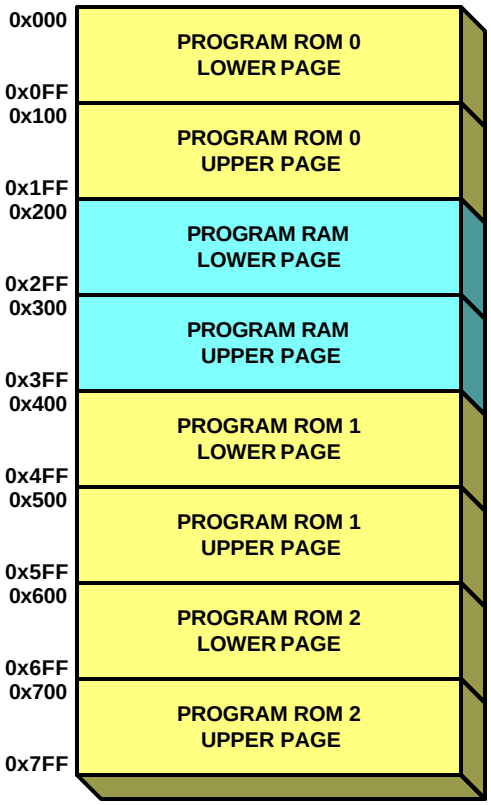



**Figure 6, MAX1780 Program Memory Organization**

## Page Boundaries

The Page Preselect bits (PA0 – PA1) of the STATUS Register determine which 512 x 12 page of program memory the MAX1780 CPU fetches instructions from. They are mapped as:

### Program Memory Selection:

| PA2 | PA1 | PA0 | Address Range | Memory Selected |
|---|---|---|---|---|
| X | 0 | 0 | 0x000 – 0x1FF | ROM (512 WORDS) |
| X | 0 | 1 | 0x200 – 0x3FF | RAM (512 WORDS) |
| X | 1 | 0 | 0x400 – 0x5FF | ROM (512 WORDS) |
| X | 1 | 1 | 0x600 – 0x7FF | ROM (512 WORDS) |

Each instruction cycle causes the Program Counter to increment, which in turn, increments the address in program memory from where instructions are fetched. Program flow within a 512 x 12 page is controlled by GOTO and CALL instructions. GOTO instructions provide 9 bits of address for the Program Counter, so program control can be transferred to any memory address within a page. The CALL instruction on the other

hand, supplies the Program Counter with only 8 bits of address. This limits jumps by the CALL instruction to the lower half of the memory page selected by the Page Preselect Bits (PA0 – PA1).

The Page Preselect Bits have no effect on program flow until a CALL or a GOTO instruction is executed. If the Program Counter is pointing to the last address of a selected memory page, when it increments it will cause the program to continue in the next higher page. However, the Page Preselect bits in the STATUS register will not be updated. Therefore, the next GOTO or CALL instruction will send the program to the page specified by the Page Preselect bits (PA1:PA0).

For example, a NOP at location 1FFh (page 0) increments the PC to 200h (page 1). A GOTO xxx at 200h will return the program to address 0xxh on page 0 (assuming that PA1:PA0 are clear). To prevent this, the page preselect bits must be updated under program control.

## Indirect Data Addressing; INDF and FSR Registers

The INDF register is not a physical register. Addressing INDF actually addresses the register whose address is contained in the FSR register (FSR is a pointer). This is indirect addressing.

The FSR is an 8-bit wide register. It is used in conjunction with the INDF register to indirectly address the data memory area. The FSR [4:0] bits are used to select data memory addresses 0x00 to 0x1F.

# Register File Organization

## Data RAM

The MAX1780 has 144 bytes of read/writable Data Memory. It is organized as follows:

8 Special Purpose Register Files (Page Independent)

8 General Purpose Registers (Page Independent)

128 General Purpose Registers (Page Dependent)

The FSR Register (bits 5 – 7) is used to select the desired data memory block.

As shown in Figure 7 below, the special purpose registers and first 8 general purpose registers (0x00 – 0x0F) are shadowed across all eight Data RAM blocks. For this reason, the contents of these registers are accessible regardless of which memory block is selected by FSR bits 5 – 7. Data stored in this area is referred to as "Page Independent" memory. Selecting the appropriate FSR bits allow access to data residing in the upper 16 bytes of each block. Data stored in this area is referred to as "Page Dependent" memory.

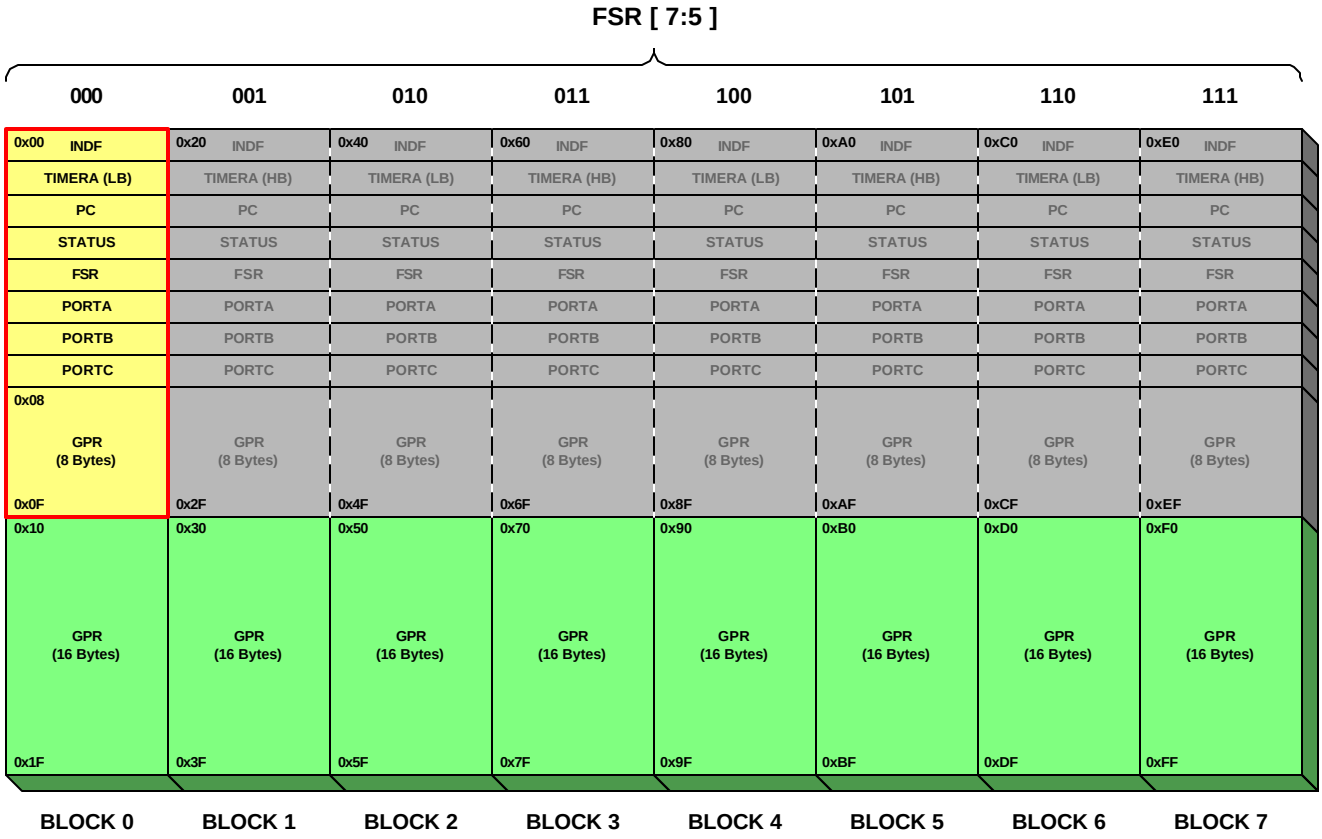



Figure 7, DRAM Organization

## File Select Register (FSR) Read/Write

The File Select Register selects which page of Data Memory can be accessed.

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| POR STATE | X | X | X | X | X | X | X | X |

| Bit7 | Bit6 | Bit5 | GPR SELECTED |
|---|---|---|---|
| 0 | 0 | 0 | GPR Page/Block 0 - TIMERA Low Byte selected. |
| 0 | 0 | 1 | GPR Page/Block 1 - TIMERA High Byte selected. |
| 0 | 1 | 0 | GPR Page/Block 2 - TIMERA Low Byte selected. |
| 0 | 1 | 1 | GPR Page/Block 3 - TIMERA High Byte selected. |
| 1 | 0 | 0 | GPR Page/Block 4 - TIMERA Low Byte selected. |
| 1 | 0 | 1 | GPR Page/Block 5 - TIMERA High Byte selected. |
| 1 | 1 | 0 | GPR Page/Block 6 - TIMERA Low Byte selected. |
| 1 | 1 | 1 | GPR Page/Block 7 - TIMERA High Byte selected. |

| Bit4 | Page Independent/Dependent Selection |
|---|---|
| 0 | Page independent Data Memory (lower 16 bytes of each Page/Block) is accessed. The lower 16 bytes of Page/Block 0 are shadowed across the other 7 page/blocks regardless of the selection of FSR bits 5 – 7. |
| 1 | Page dependent Data Memory (upper 16 bytes of each Page/Block) is accessed. Selecting bits 5 – 7 of the FSR accesses a Page/Block. Selecting FSR bits 0 - 3, accesses individual bytes within a Page/Block. |

Selection of Register File or Data RAM

| Bit3 | Bit2 | Bit1 | Bit0 | Data RAM Location |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0x00 |
| 0 | 0 | 0 | 1 | 0x01 |
| 0 | 0 | 1 | 0 | 0x02 |
| 0 | 0 | 1 | 1 | 0x03 |
| 0 | 1 | 0 | 0 | 0x04 |
| 0 | 1 | 0 | 1 | 0x05 |
| 0 | 1 | 1 | 0 | 0x06 |
| 0 | 1 | 1 | 1 | 0x07 |
| 1 | 0 | 0 | 0 | 0x08 |
| 1 | 0 | 0 | 1 | 0x09 |
| 1 | 0 | 1 | 0 | 0x0A |
| 1 | 0 | 1 | 1 | 0x0B |
| 1 | 1 | 0 | 0 | 0x0C |
| 1 | 1 | 0 | 1 | 0x0D |
| 1 | 1 | 1 | 0 | 0x0E |
| 1 | 1 | 1 | 1 | 0x0F |

## Status Register (Read/Write)

The Status Register contains the arithmetic status of the ALU, TMRB overflow status, the global interrupt enable/disable bit, and the Page Preselect bits for selecting from which page of program memory program code will be fetched.

The Status Register can be the destination for any instruction. If the Status Register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. It is recommended, therefore, that only BCF, BSF and MOVWF instructions be used to alter the STATUS register because these instructions do not affect the Z, DC or C bits from the STATUS register.

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | **PA2** | **PA1** | **PA0** | **TMRBF** | **INTOFF** | **Z** | **DC** | **C** |
| POR STATE | 0 | 0 | 0 | 0 | 1 | X | X | X |

| PA2 | PA1 | PA0 | Memory Range Selected | Type | Length |
|---|---|---|---|---|---|
| X | 0 | 0 | 0x000 – 0x1FF | ROM | 512 WORDS |
| X | 0 | 1 | 0x200 – 0x3FF | RAM | 512 WORDS |
| X | 1 | 0 | 0x400 – 0x5FF | ROM | 512 WORDS |
| X | 1 | 1 | 0x600 – 0x7FF | ROM | 512 WORDS |

Note: PA2 is not implemented but can be used as storage.

| TMRBF | |
|---|---|
| 0 | TIMER B overflow flag CLEARED |
| 1 | TIMER B overflow flag SET |

| INTOFF | |
|---|---|
| 0 | Enable Global Interrupts |
| 1 | Disable Global Interrupts |

| Z | |
|---|---|
| 0 | Arithmetic or Logic Operation Results is NOT Zero |
| 1 | Arithmetic or Logic Operation Results is Zero |

| DC | |
|---|---|
| 0 | No CARRY from Bit3 to Bit4 |
| 1 | CARRY from Bit3 to Bit4 |

Note: Effected only by ADDWF & SUBWF Instructions

| C | |
|---|---|
| 0 | No CARRY from Bit7 (MSB). When no CARRY generated in Addition, or when subtraction result is negative. |
| 1 | CARRY from Bit7 (MSB). When CARRY is generated in Addition, or when subtraction result is zero or positive. |

# Modes Of Operation

The MAX1780 has four modes of operation:

1. Shutdown ($\overline{SHDN}$ = Low)
2. Sleep
3. Master Clear ($\overline{MCLR}$ = Low)
4. Operate (Program Execution)

## Shutdown

In shutdown mode, the MAX1780 consumes practically no current. This mode is useful for reducing the self-discharge of battery packs in transit to customers or being stored for long periods.

### Entering Shutdown

To enter shutdown mode, the $\overline{SHDN}$ pin should be grounded. When the voltage on the $\overline{SHDN}$ pin falls below 0.4V, the MAX1780 will enter a very low power (typically 1nA) consumption mode.

### Recovering From Shutdown

To recover from shutdown mode, the $\overline{SHDN}$ pin should be connected to the BATT pin through a 3Megohm resistor. After the voltage at the $\overline{SHDN}$ pin has risen to a level greater than 2.2V, the MAX1780 will Power On Reset (POR) and begin to execute code.

## Sleep

In sleep mode, instruction execution is suspended. The internal instruction oscillator may be on or off in sleep mode depending on the contents of the OSLB bit (D7) of the Option Register. The internal instruction oscillator turns on whenever sleep mode is exited regardless of the condition of the OSLB bit.

### Entering Sleep Mode

Executing a SLEEP instruction enters sleep Mode.

### Wake-Up From Sleep Mode

The MAX1780 can wake up from SLEEP through one of the following events:

1. An external reset input on the $\overline{MCLR}$ pin.
2. A Watchdog time-out.
3. An interrupt from any enabled source will wake up the MAX1780, regardless of the state of the INTOFF bit in the STATUS Register.

## Master Clear

Placing a logic level LOW on the $\overline{MCLR}$ pin will cause the MAX1780 to reset all its internal logic circuitry. While $\overline{MCLR}$ is LOW all program execution is halted. Additionally, all GPIO pins are high-impedance, and the CHG and DIS pin drivers will rise to VBATT, opening both pack protection MOSFETs.

Releasing the logic LOW on the $\overline{MCLR}$ pin, allowing it to rise to VDD, will allow the processor core to boot and begin program execution.

## Operate (Program Execution)

Whenever instructions are executing, the MAX1780 is in operate mode.

## Analog And Mixed Signal Peripheral Interface

The MAX1780 integrates an arsenal of analog and mixed signal peripheral devices. The processor core communicates with these on-board peripherals through PORT B, as shown in Figure 8.

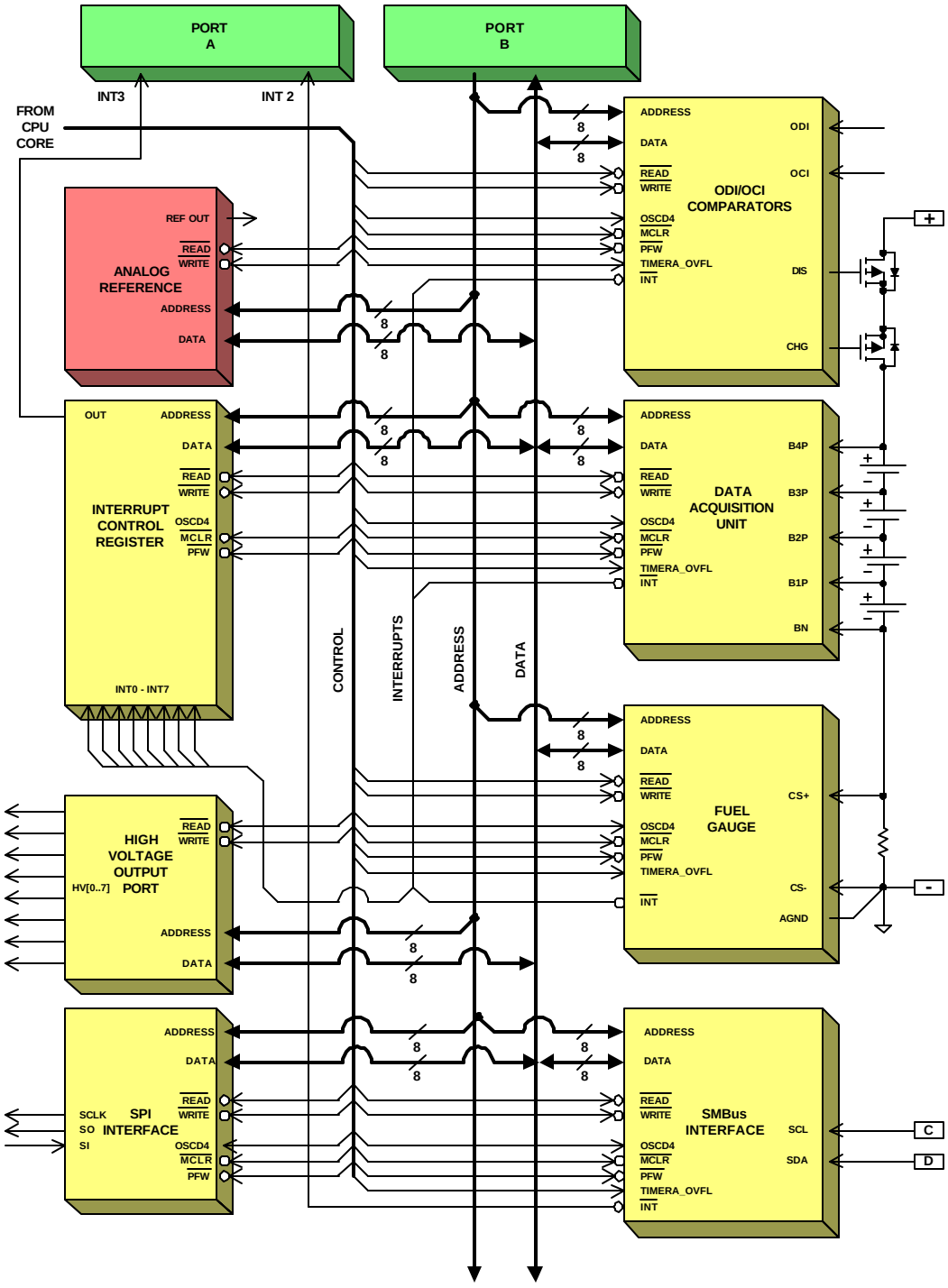



**Figure 8, MAX1780 On-Board Peripherals**

## MAX1780 Power Supply Sequencing

The MAX1780 incorporates both analog and digital circuitry, which each have an optimal power supply range. The digital circuitry can begin operating with VAA voltages as low as 2.7V, whereas the analog circuitry begins to operate at 3.0V. This means that the MAX1780 CPU can begin executing program code before it's analog peripherals have powered up.

The MAX1780 uses two internal control signals, $\overline{POR}$ for the digital circuitry, and $\overline{PFW}$ for the analog circuitry, to determine if the VAA supply has reached a voltage level high enough for both the digital and analog circuitry to operate. Figure 9 illustrates the MAX1780 power supply sequencing.

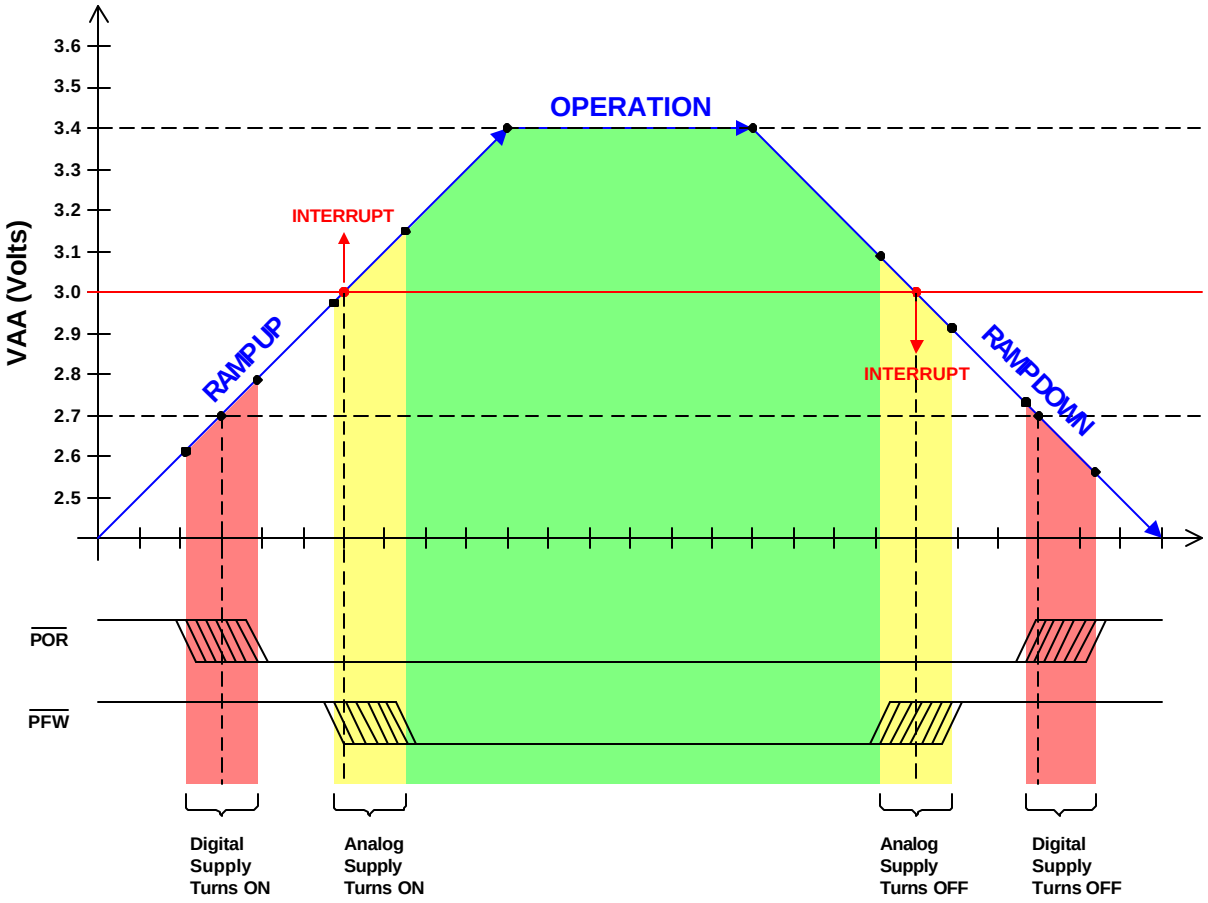



Figure 9, MAX1780 Power Supply Sequencing

Whenever the VAA supply rises above, or falls below 3.0V an interrupt is generated. Please refer to the Peripheral Interrupt Control Register section for a detailed description of this interrupt. Immediately after power-up, the interrupt can be used to indicate that both the digital and analog supplies have reached a reliable voltage level for operation. Any interrupts that occur after the initial power-up sequence, could be an indication that the VAA supply has either momentarily fallen below 3.0V, or is powering off.

## MAX1780 Special Purpose Port Registers

Each port line can be individually programmed as an input or output. This is done by using a special purpose instruction TRIS, which matches a bit pattern with the port lines. Writing a '0' to a port line's TRIS Register configures this port line as an output. Conversely, setting a port line's TRIS Register to '1' configures the port line as an input.

## PORTA

The MAX1780 PORTA[3:0] signal lines are not connected to external pins. The PORTA data latch is however read/writable, so bits 0 – 3 can be freely used as storage.

## PORTA Data Latch (Read/Write)

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | **IF3** | **IF2** | **IF1** | **TMRAF** | **PORTA3** | **PORTA2** | **PORTA1** | **PORTA0** |
| POR STATE | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

IF3, IF2, IF1 are the Interrupt Flags for INT3, INT2, INT1 respectively. They will be set whenever there is a respective interrupt.

TMRAF flag is set when there is a TIMERA overflow.

## PORTA TRIS Latch (Write Only)

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | **MA3** | **MA2** | **MA1** | **MTMRA** | **X** | **X** | **X** | **X** |
| POR STATE | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

MA3, MA2, MA1, MTMRA are the interrupts of INT3, INT2, INT1, TIMERA Enable Control. Writing a '0' to an individual bit will enable the respective interrupt.

## PORTB

PORTB is a special purpose port used to communicate with the on-chip peripherals. The address for a desired peripheral is written to the PORTB TRIS Latch, and the data is either written to or read from the PORTB Data Latch using a MOVF instruction.

Care should be taken to properly protect Main Loop PORTB peripheral transactions from interrupt service routines that may also use PORT to communicate to peripherals.

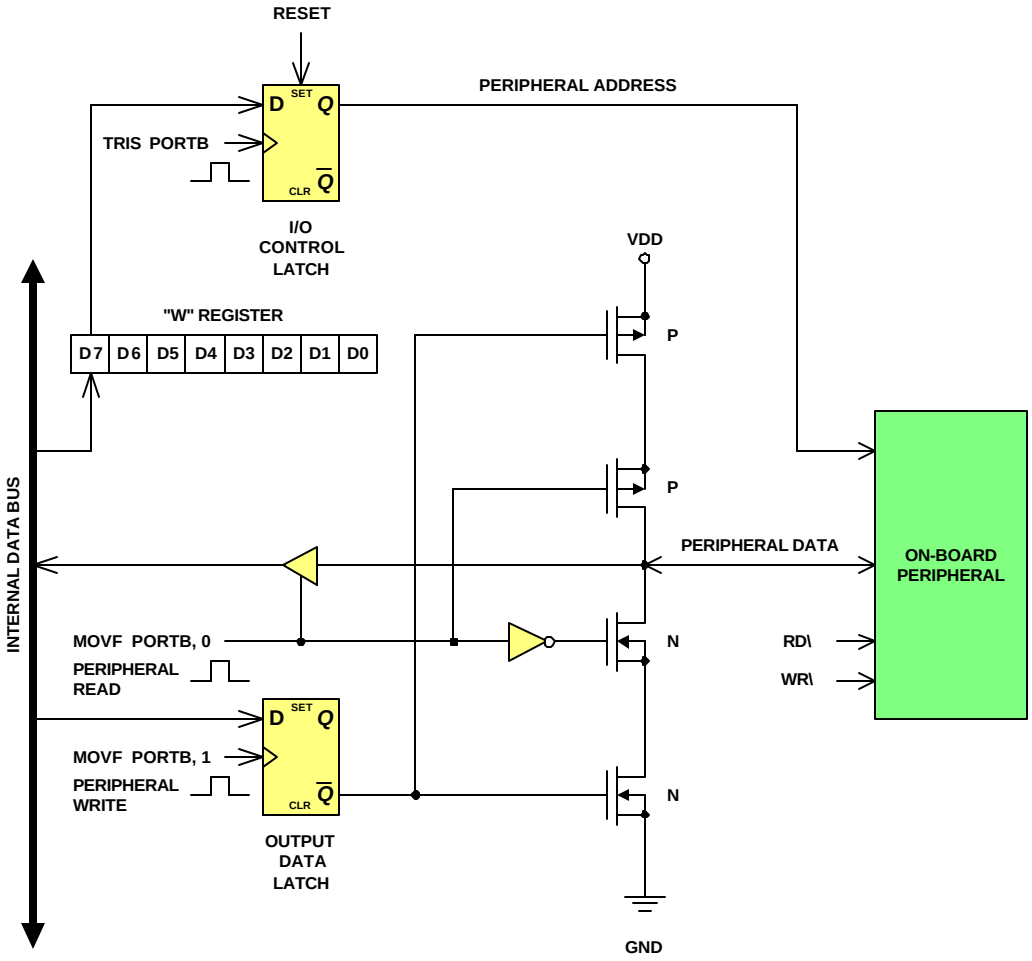



Figure 10, PORT B Structure (1 Bit)

## PORTB Data Latch (Read/Write)

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| POR STATE | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## PORTB TRIS Latch (Write Only)

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | **A7** | **A6** | **A5** | **A4** | **A3** | **A2** | **A1** | **A0** |
| POR STATE | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

# Accessing The On-board Peripherals

The MAX1780 processor core can access on-board peripherals provided that the Setupint Register PADDR bit D0 is set to '1'. The peripheral's address is first written to the PORT B TRIS latch. If a peripheral read is desired, then a MOVF instruction is executed to latch the peripheral bus data into the "W" Register. If a peripheral write is wished, then in addition to writing the peripheral's address to the PORT B TRIS latch, the data for the peripheral must be written to the PORT B DATA latch. The following code example shows how to read data from an on-board peripheral using the PORT B internal interface.

Peripheral Read

```
MOVLW   ADDRESS     Load the peripheral's address in W register.
TRIS    Port_B      Latches the address.
MOVF    Port_B,0    Generates Read signal and returns peripheral INPUT data to the W register.
```

Programming code for write operations is similar, except that two extra move instructions are required to latch the data sent to the PORT B peripheral.

Peripheral Write

```
MOVLW   ADDRESS     Load the peripheral's address in W register.
TRIS    Port_B      Latches the address.
MOVLW   DATA        Load Peripheral's data in W register.
MOVWF   Port_B      Puts Data Into B For Output.
MOVF    Port_B,1    Generates Write signal and sends OUTPUT data to the peripheral.
```

## PORTC (IO0 – IO7)

Port C shares duty as a general-purpose I/O (GPIO) port and SPI interface. Bits IO0, IO1, and IO2 are used exclusively in normal operation by the high-speed SPI interface. Bit IO3, under software control, is used for the SPI CS\ signal. Bit IO4, when configured as an input, can be used to increment Timer A. Bit IO7 shares use as an external input to the INT1 interrupt. Bits IO5 and IO6 are free GPIOs.

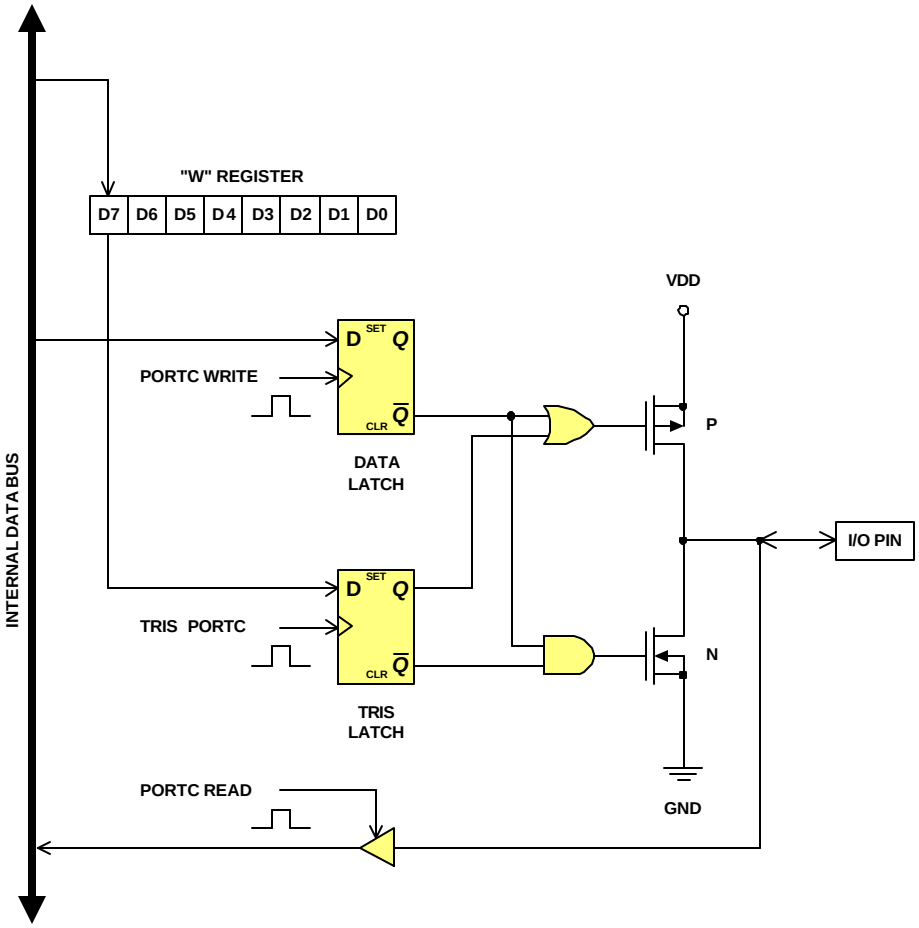



Figure 11, PORTC GPIO Structure (1-Bit)

## PORTC GPIO Operation

The equivalent circuit for one of the PORTC GPIO port pins is shown in Figure 11. PORTC, IO0 - IO7 may be used for both input and output operations. For input operations PORTC is non-latching. Any input must be present until read by an input instruction (e.g., MOVF PORTC, W). The outputs are latched and remain unchanged until the output latch is rewritten. To use a PORTC I/O pin as output, the corresponding direction control bit in TRIS latch must be cleared. For use as an input, the corresponding TRIS bit must be set. Any I/O pin can be programmed individually as input or output.

The TRIS latch is loaded with the contents of the “W” Register by executing the TRIS f instruction. A '1' from a TRIS register bit puts the corresponding output driver in a hi-impedance mode. A '0' puts the contents of the output data latch on the selected pins, enabling the output buffer. The TRIS registers are “write-only” and are set (output drivers disabled) upon RESET.

Note: A read of the ports reads the pins, not the output data latches. That is, if an output driver on a pin is enabled and driven high, but the external system is holding it low, a read of the port will indicate that the pin is low.

At power-on-reset, all port lines are tri-stated. All unused port lines should be tied to VDD. Please refer to the MAX1780 Electrical Characteristics and Absolute Maximum Ratings when connecting the I/O port lines to external circuitry.

## I/O Programming Considerations

Some instructions operate internally as read followed by write operations. The BCF and BSF instructions, for example, read the entire port into the CPU, execute the bit operation and re-write the result. Caution must be used when these instructions are applied to a port where one or more pins are used as input/outputs. For example, a BSF operation on bit-7 of PORTC will cause all eight bits of PORTC to be read into the CPU, bit-7 to be set and the PORTC value to be written to the output latches. If another bit of PORTC is used as a bi-directional I/O pin (say bit0) and it is defined as an input at this time, the input signal present on the pin itself would be read into the CPU and rewritten to the data latch of this particular pin, overwriting the previous content.

For this reason, it is good programming practice to "shadow" the PORTC data latch. This involves maintaining a copy of the PORTC data latch contents in a data RAM location. Individual bits are set by first "OR-ing" the desired bits with the PORTC data latch copy, and then moving the latch copy to the PORTC latch. Likewise, individual port bits can be cleared by "AND-ing" the reciprocal of the desired bits with the PORTC data latch copy, followed by a move to the data latch.

## PORTC Data Latch

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | **IO7 INT1** | **IO6** | **IO5** | **IO4/ TMRA** | **IO3 (CS\)** | **IO2/ SI** | **IO1/ SO** | **IO0/ SCLK** |
| POR STATE | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## PORTC TRIS Latch

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| | **TRIS7** | **TRIS6** | **TRIS5** | **TRIS4** | **TRIS3** | **TRIS2** | **TRIS1** | **TRIS0** |
| POR STATE | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## Timers And Watchdog

Figure 12 below shows the organization of Timer A, Timer B, and the Watchdog Timeout circuitry.

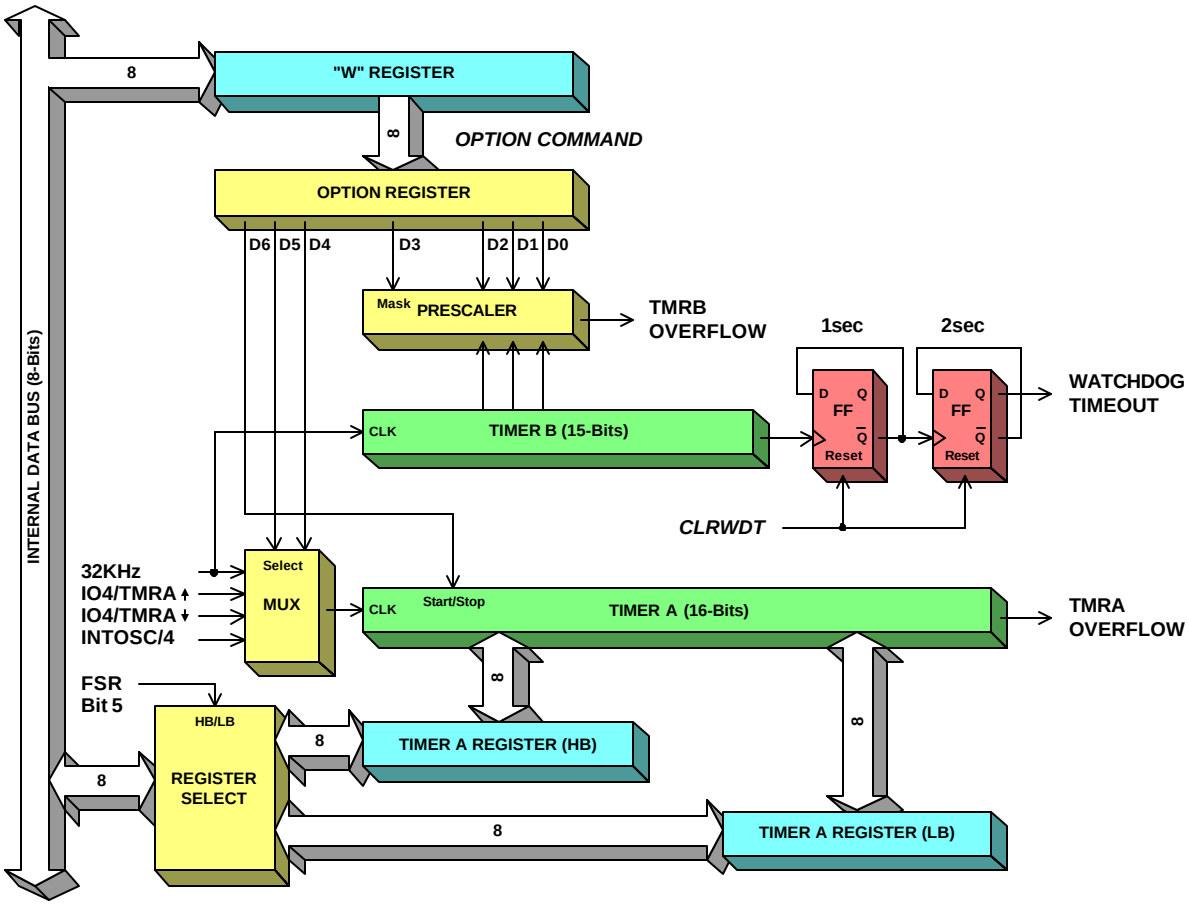



Figure 12, MAX1780 Timers

## Timer A (TMRA)

TMRA is a general-purpose 16-bit ripple counter that can be configured to use one of four clock sources, IO4/TMRA pin rising edge, IO4/TMRA pin falling edge, 32KHz oscillator rising edge, or the rising edge of the instruction oscillator divided by four. Setting the TAS0 and TAS1 bits in the Option Register selects the clock source. When either the 32KHz oscillator or IO4/TMRA pin clock the timer, the timer operates asynchronously of the MAX1780 processor core. This means that the timer can count events even when the CPU core is in sleep mode, as it does not need the instruction oscillator to count. The user can enable Timer A by clearing the TMRAD bit in the Option Register. To disable Timer A, set the TMRAD bit in the Option Register. When TMRA overflows, the ITMRA flag in the Status Register is set and an interrupt will be generated. This flag should be cleared by the TMRA interrupt service routine. If the MAX1780 CPU is in SLEEP mode when a TMRA overflow occurs, it will wake-up the CPU.

## Timer B (TMRB)

Timer B is a 15-bit ripple counter permanently attached to the 32KHz crystal oscillator. It has a 3-bit prescaler to divide the oscillator down to obtain timer periods between 15.625msec and 1sec. The prescaler can be adjusted by writing to bits PS0 through PS2 in the Option Register and executing an OPTION instruction. If the MSKB bit in the Option Register is cleared, each time the TMRB period is exceeded (overflows), an interrupt is generated and the TMRBF flag in the Status Register is set. The TMRB interrupt service routine should first,

execute a CLRTI instruction and then clear the TMRBF flag. If the MAX1780 CPU is in SLEEP mode when a TMRB overflow occurs, it will wake-up the CPU. TMRB is a logical choice for creating an accurate real-time clock that generates recurring interrupts at a desired period.

## Watchdog Timer (WDT)

The primary function of the Watchdog Timer is to be a failsafe method for recovering from programs that are stuck in an endless loop, or may have inadvertently corrupted the stack. When a Watchdog timeout occurs, the MAX1780 microcontroller core is reset and the entire boot-up process is repeated. Figure 12 shows the Watchdog Timer and how it is derived from Timer B. Timer B is a free running 16-bit ripple counter, and is clocked by the 32KHz oscillator. Timer B reaches its maximum count each second, and will overflow, setting the flip-flop connected to its output. If the flip-flop is not cleared, by executing a CLRWDT instruction, by the time Timer B overflows again (2 seconds), the second flip-flop toggles, resulting in a reset of the MAX1780. The Watchdog timeout is held off or cleared by executing a CLRWDT instruction before the Watchdog's timer period has expired. Good programming practice will insure that CLRWDT instructions are properly distributed to prevent a Watchdog timeout in normal operation. A word of caution, executing the SLEEP or OPTION instructions does NOT clear the watchdog timeout.

# Interrupts

## Description

The MAX1780 CPU interrupt structure is depicted in Figure 13 below. All interrupts vector program execution to memory address 0x0200. All interrupts have the same priority. The first interrupt input (INT1) is connected to pin IO7, and may be used by external circuitry. The second interrupt input (INT2), is used exclusively by the SMBus Interface. Within the SMBus Interface, there are multiple interrupt sources. Please refer to the SMBus section of this document for details. The third interrupt input (INT3), is driven by the Peripheral Interrupt Controller. This block has mask and status registers for eight peripheral interrupt sources. Please refer to the Peripheral interrupt Controller section for details.

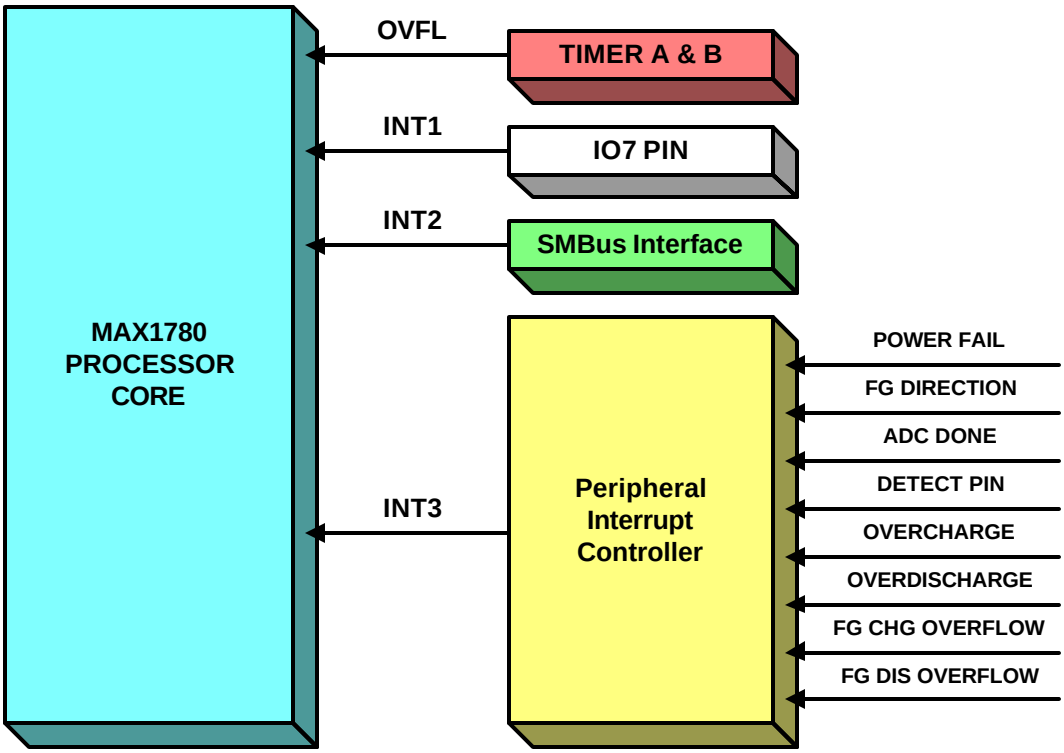



**Figure 13, MAX1780 Interrupt Structure**

## Peripheral Interrupt Control Registers

The Peripheral Interrupt Control registers INTSTAT and INTREN funnel all of the on board peripheral interrupt sources to INT3 in the MAX1780 core. The only exceptions are the interrupts generated by the SMBus Interface, which are routed to INT2. The INTSTAT Register latches the interrupt events so that they can provide status, which can be read at any time. The INTREN Register is provided to enable or disable interrupts to INT3. Please note that even though a particular interrupt can be disabled (masked) by clearing the appropriate bit in the INTREN Register, the associated interrupt flag in the INTSTAT Register will still be set when the interrupt occurs. Figure 14 shows the INSTAT and INTREN registers, and how they interact to provide interrupt control and status.

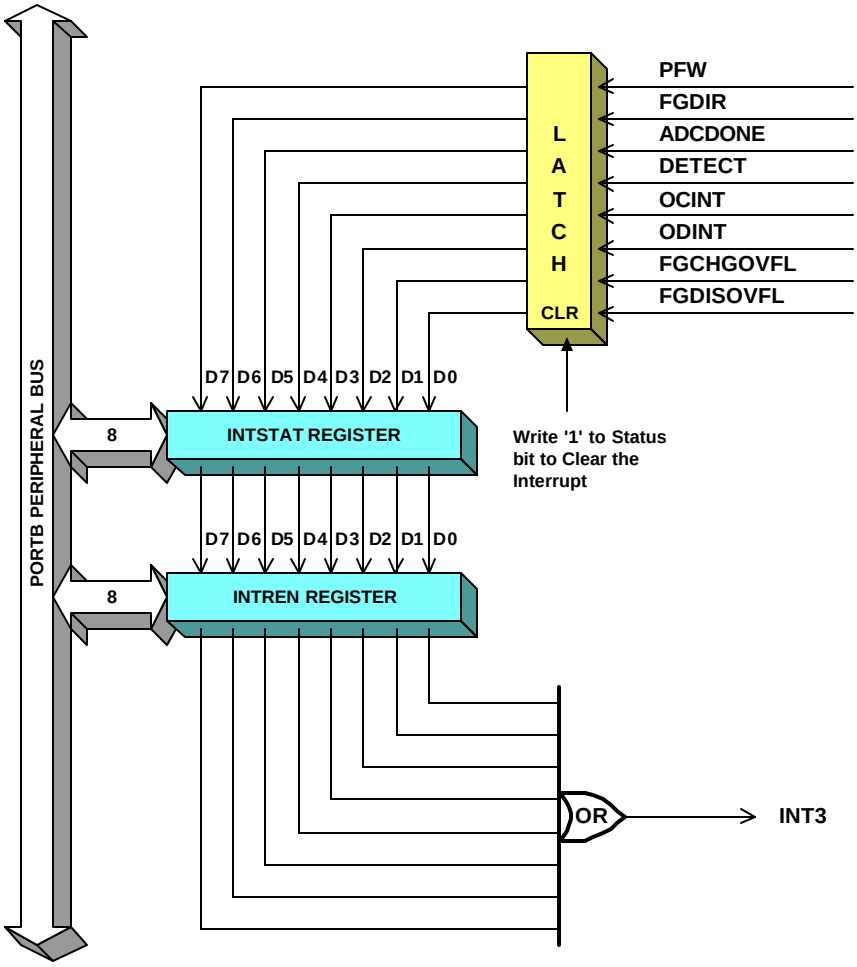



Figure 14, Peripheral Interrupt Control Registers

## Interrupt Status Register (INTSTAT) Operation

The Interrupt Status Register latches interrupt events from 8 peripheral sources. When a peripheral interrupt occurs, the corresponding bit in the INTRSTAT Register will be set. It can be read through the PORTB Interface any number of times without affecting the state of the status flags. This is true regardless of the interrupt enable register settings. Writing a '1' to a particular bit in the interrupt status register will clear the interrupt status corresponding to that bit.

## Interrupt Enable Register (INTREN) Operation

The Interrupt Enable Register (INTREN) controls which peripheral interrupt sources will trigger Interrupt 3 (INT3) in the MAX1780 CPU. The register values can be read through the PORTB Interface any number of times without affecting their state. The Interrupt Enable Register bits will be cleared to '0' whenever $\overline{MCLR}$ is asserted low or a power on reset (POR) occurs. Clearing the bits in this register does not prevent the INTSTAT Register from latching peripheral interrupts that occur, this only masks them from INT3.

## Interrupt Control Register Descriptions

### INTSTAT Register (Read/Write): PORTB Address = 0x0E

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | PFW | 0 | Power Fail Warning. VAA Supply > 3V After POR, Or Has Fallen Below 3V During Operation. |
| D6 | FGDIR | 0 | Fuel Gauge Direction Change. |
| D5 | ADC | 0 | ADC conversion start and completion. |
| D4 | DETECT | 0 | A Rising Or Falling Edge On The DETECT Pin Has Occurred. |
| D3 | OCINT | 0 | Charge Current Limit Exceeded. |
| D2 | ODINT | 0 | Discharge Current Limit Exceeded. |
| D1 | FGCHGOVFL | 0 | Fuel Gauge Charge Counter Overflow. |
| D0 | FGDISOVFL | 0 | Fuel Gauge Discharge Counter Overflow. |

Note: The register is read to determine the interrupt status. To clear an interrupt, write a '1' to the bit that corresponds to the interrupt.

### INTREN Register (Read/Write): PORTB Address = 0x0F

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | PFWMSK | 0 | 0 – PFW Interrupt Masked.<br>1 – PFW Interrupt Enabled. |
| D6 | FGDIRMSK | 0 | 0 – FGDIR Interrupt Masked.<br>1 – FGDIR Interrupt Enabled. |
| D5 | ADCMSK | 0 | 0 – ADC Interrupt Masked.<br>1 – ADC Interrupt Enabled. |
| D4 | DETMSK | 0 | 0 – DETECT Interrupt Masked.<br>1 – DETECT Interrupt Enabled. |
| D3 | OCIMSK | 0 | 0 – OCINT Interrupt Masked.<br>1 – OCINT Interrupt Enabled. |
| D2 | ODIMSK | 0 | 0 – ODINT Interrupt Masked.<br>1 – ODINT Interrupt Enabled. |
| D1 | FGCHGMSK | 0 | 0 – FGCHGOVFL Interrupt Masked.<br>1 – FGCHGOVFL Interrupt Enabled. |
| D0 | FGDISMSK | 0 | 0 – FGDISOVFL Interrupt Masked.<br>1 – FGDISOVFL Interrupt Enabled. |

# Analog Peripherals

## 3.5MHz Instruction Oscillator

The MAX1780 contains an internal instruction execution oscillator that does not require an external crystal for operation. It is factory trimmed to 3.5MHz. In sleep mode the internal instruction oscillator can be turned off to save power. The internal oscillator is guaranteed to start up within 2 µs, minimizing interrupt latency. Any interrupt automatically exits sleep mode. The OSLB bit in the OPTION register controls whether or not the internal instruction oscillator is turned off in sleep mode. The internal instruction oscillator turns on whenever sleep mode is exited regardless of the condition of the OSLB bit.

## 32KHz Oscillator

The 32KHz oscillator is used by several of the MAX1780 peripherals. It provides input clocks for TIMERA, TIMERB, the Data Acquisition Unit, and the Fuel Gauge. The oscillator requires only an external 32.768KHz watch crystal for proper operation. The 32KHz Oscillator is a Pierce-type crystal oscillator in which the output frequency is tuned by varying the total capacitance across the crystal's terminals. This capacitance is referred to as the "Load Capacitance" $C_L$ on crystal datasheets, and can vary depending on the manufacturer. Load Capacitance is calculated as follows:

$$\mathbf{C_L = C_{OSC} + C_{PCB} + C_0}$$

Where $C_{OSC}$ is the MAX1780's on-chip load capacitance (2.5pF typical), $C_{PCB}$ is PCB layout parasitic capacitance, and $C_0$ is the shunt capacitance of the 32KHz watch crystal.

For reliable oscillator start-up under worst-case conditions, insure that $C_L$ is less than 7pF. We do not recommend adding external capacitance to tune the oscillator frequency. Use caution when connecting an oscilloscope probe to OSC1, the 10pF scope probe capacitance is large enough to stop the 32KHz oscillator.

## Low Drop Out Linear Regulator

The Low Drop Out Linear Regulator Unit regulates a 4V to 28V DC input voltage on the BATT pin, down to 3.4V. The 3.4V supplies the MAX1780 internal circuitry, and is available for external circuitry on the VAA output pin. Please note that for proper operation, the VAA and VDD pins must be connected as close as possible to the chip. The VAA output can supply 3.4V to external loads at up to 10mA. The Linear Regulator's current limit is typically 40mA. The VAA output should be bypassed with a 0.47µF capacitor to ground.

## Precision Bandgap Reference

The Precision Bandgap Reference provides 1.217V to the Data Acquisition Block. It is used internally and not brought out to a pin. The MAX1780 powers up with this reference shutdown and before attempting to make any measurements with the Data Acquisition Unit or Fuel Gauge, the user must write a '1' to bit D7 of the REFCONFIG Register.

**REFCONFIG Register (Write Only): PORTB Address = 0x09**

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | REFON | 0 | 0 = Reference OFF<br>1 = Reference ON |
| D6 | RFU | - | Data bit 6 |
| D5 | RFU | - | Data bit 5 |
| D4 | RFU | - | Data bit 4 |
| D3 | RFU | - | Data bit 3 |
| D2 | RFU | - | Data bit 2 |
| D2 | RFU | - | Data bit 1 |
| D2 | RFU | - | Data bit 0 |

# Mixed Signal Peripherals

## Fuel Gauge Unit

### General Description

The Fuel Gauge measures the cumulative charge into (charging) and out of (discharging) the system battery pack and stores the information in one of two internal, independent charge and discharge counters. The unit also informs the host of changes in the direction of current flow. Communication with the Fuel Gauge is via Port B, and allows access to charge/discharge counters and internal registers.

### Features

- True Coulomb Counting, Integrating Fuel Gauge
- Separate 16-bit Charge and Discharge Counters
- Counter Overflow and Current Direction Change Interrupts
- 4 Counter Latching Sources
- Automatic Cancellation Of Input Offset Voltage

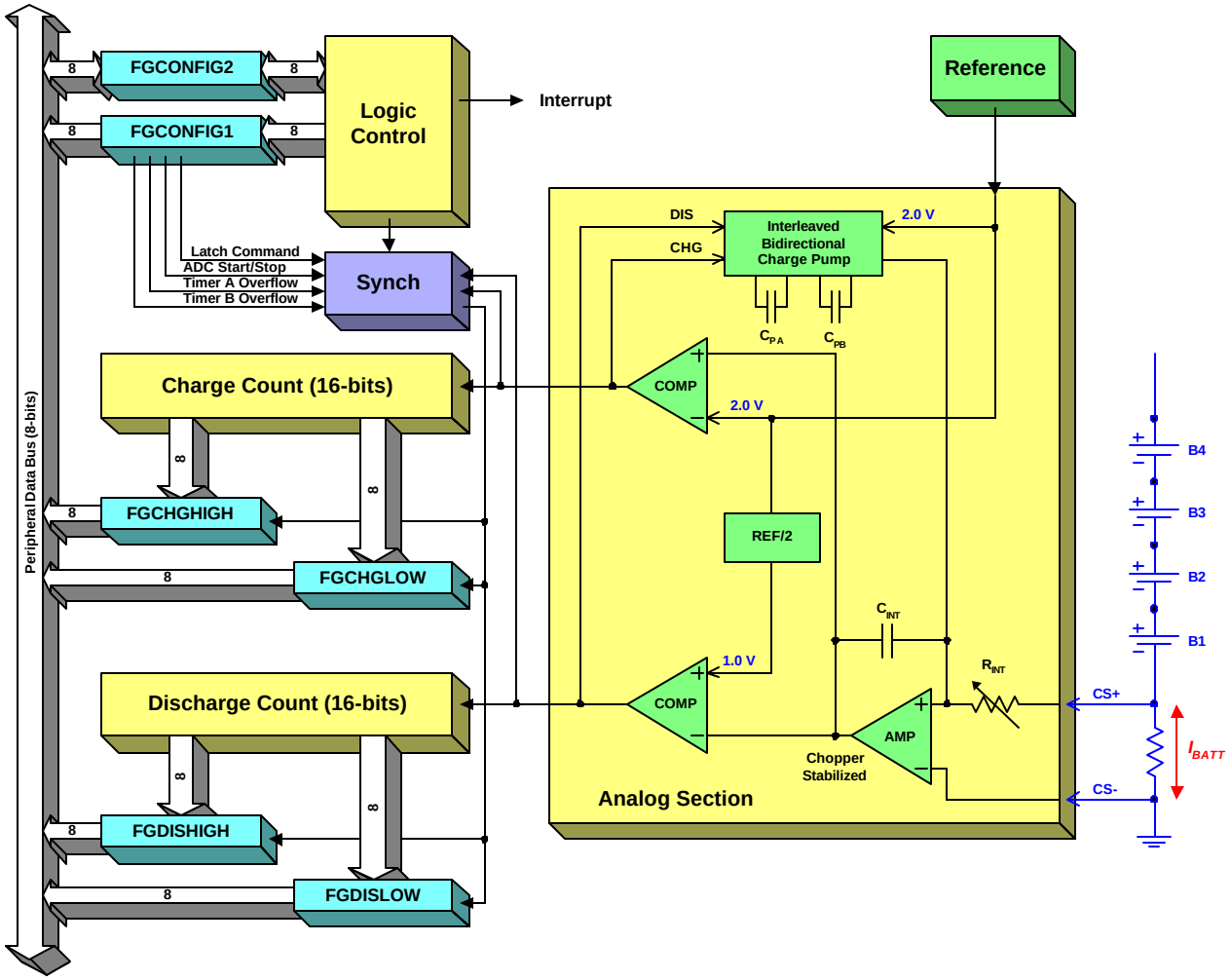



Figure 15, Fuel Gauge Block Diagram

## Automatic Cancellation Of Input Offset Voltage

The MAX1780 Fuel Gauge uses a Chopper-Stabilized amplifier to couple the voltage across the current sense resistor to the Voltage-To-Frequency converter. This voltage, which can be a high as +/- 137.33mV at heavy load currents, can be only a few microvolts at light loads. Therefore, the amount of input offset voltage determines the minimum current sensitivity of the fuel gauge. The Chopper-Stabilized amplifier continually samples and corrects for its inherent offset. This cancellation occurs in conjunction with the Fuel Gauge's Coulomb counting circuitry. While the circuit is performing Fuel Gauge measurements, it is continually canceling the internal input offset voltage. The MAX1780 Fuel Gauge therefore, requires no software calibration. With careful attention to PCB layout, input offsets of less than 1µV can be expected.

## Coulomb Counting

The Fuel Gauge's Coulomb counting circuit monitors the differential voltage present at the CS+ and CS- pins. In a typical application, the CS+ and CS- pins are connected across a sense resistor that is in series with the battery pack cells. This voltage is converted to a frequency proportional to the rate at which the current is flowing through the sense resistor, and the circuit counts Coulombs of charge by incrementing either the Charge Counter or the Discharge Counter accordingly.

## Charge And Discharge Counters

Figure 15 shows the functional diagram of the Fuel Gauge's Coulomb-counter section. The Coulomb counter's output increments (but never decrements) one of two independent 16-bit counters: Charge Count for charging currents, and Discharge Count for discharging currents. By independently counting the charge and discharge currents, the Fuel Gauge can accommodate any algorithm to account for battery pack energy-conversion efficiency. The 16-bit Charge and Discharge Count latch registers are each divided into 2 bytes: FGCHGLOW, FGCHGHIGH, FGDISLOW, and FGDISHIGH. See the Fuel Gauge Register Descriptions for details of the different registers. Charge Count and Discharge Count reset to zero whenever a power-on reset executes, or when the configuration word's FGCLRCHG FGCLRDIS bits are set. Use of the FGCLRCHG and FGCLRDIS bits to clear the fuel gauge counters is not recommended as part of a fuel gauging algorithm, as it is possible to loose counts during the clear operation. Each counter also resets any time an overflow condition occurs. When a counter overflows, it simply clears and begins counting from 0. Interrupts are generated on counter overflows unless they are masked by the FGCHGOVFL and FGDISOVFL bits in the Interrupt Controller INTREN register.

Writing a one to the FGLATCHNOW bit in the Fuel Gauge's FGCONFIG2 Register latches the instantaneous counts for both the Charge and Discharge Counters without clearing the counters. The 16-bit charge count value can be obtained by reading the data in the FGCHLOW and FGCHGHIGH latch registers. Similarly, the Discharge Count can be obtained by reading the data in the FGDISLOW and FGDISHIGH registers.

The gain factor is the constant of proportionality that relates the counter values stored in the Charge Count and Discharge Count registers to the amount of charge flow into or out of the battery pack. The electrical characteristics table specifies the maximum v-to-f converter frequency and the full-scale sense resistor voltage. With these two values the gain factor ($FG_{GAIN}$) can be calculated as follows:

Determining Fuel Gauge Gain:

$$FG_{GAIN} = \frac{50 \cdot KHz}{137.33 \cdot mV} \qquad FG_{GAIN} = 3.641 \times 10^5 \frac{Hz}{V}$$

Multiplying the sense resistor value by the Gain Factor, the number of counter increments generated per Coulomb can be determined.

Determining Fuel Gauge Count Rate:

Given: $R_{CS} = 0.020 \cdot \Omega$

$$\text{Count}_{RATE} = R_{CS} \cdot FG_{GAIN} \quad \text{Count}_{RATE} = 7.282 \times 10^3 \frac{\text{Count}}{\text{C}}$$

Therefore, a 20mΩ current-sense resistor sets up a counter rate of 7.28 x $10^3$ counts per Coulomb. A higher conversion gain (larger $R_{CS}$) increases resolution at low currents, but limits the maximum measurable current. Likewise, a smaller conversion gain (smaller $R_{CS}$) decreases resolution at low currents, but increases the maximum measurable current. This provides a good balance between resolution and input current range for many applications.

Calculating The Fuel Gauge "Bucket" Size:

Given: $R_{CS} = 0.020 \cdot \Omega$

$$FG_{BUCKET\_SIZE} = \frac{1}{\text{Count}_{RATE}} \cdot \frac{1 \cdot \text{hr}}{3600 \cdot \text{sec}} \cdot \frac{1000 \cdot \text{mA}}{1 \cdot \text{A}} \cdot \frac{65536 \cdot \text{Count}}{1 \cdot \text{Overflow}}$$

$$FG_{BUCKET\_SIZE} = 2.5 \frac{\text{mA} \cdot \text{hr}}{\text{Overflow}}$$

The fuel gauge “bucket” size, which is the amount of energy necessary to overflow either the charge or discharge counters, is an integral factor in calculating a battery’s remaining capacity. Knowing the amount of charge or discharge energy at each overflow interrupt, in milliamp-hours, makes updating the calculated battery remaining capacity as simple as adding and subtracting. In the example above, with $R_{CS}$ equal to 0.020Ω, the fuel gauge charge or discharge counters will trigger an interrupt whenever 2.5 mAh of energy has flowed into or out of the battery.

## Current Direction Change Detection Function

The Fuel Gauge’s direction-change detection function informs the host whenever the current flow changes direction. The direction-change function is simple: the FGCONFIG1 Register FGCHGSTAT bit is set to 1 when the voltage potential across CS+ and CS- is positive. When the voltage from CS+ to CS- is negative (discharge) this status bit is set to 0. The FGCHGSTAT bit is READ ONLY. The INTSTAT Register, in the peripheral interrupt controller, has a dual edge triggered input for the FGCHGSTAT bit called FGDIR. FGDIR sets anytime there is a change in current flow direction on the sense resistor. This is useful for interrupting the CPU for routines in which the host must be informed immediately of a change in current-flow direction. To enable this interrupt set the FGDIR bit in the INTREN Register to a ‘1’ (See the section on Peripheral Interrupt Controller).

## Counter Latching Source And Arbiter

The FGMUX bits in the FGCONFIG1 Register select one of four sources for latching the Fuel Gauge’s charge and discharge counter values.

### Direct CPU Control

The fuel gauge charge and discharge counter values can be latched under direct CPU control by first setting the FGMUX bits D4 and D5 to ‘00’ in the FGCONFIG1 Register, and then writing a ‘1’ to the FGLATCHNOW bit D7 in the FGCONFIG2 Register. This action essentially takes a “snapshot” of the instantaneous counter values

and stores them in the FGDISHIGH, FGDISLOW, FGCHGHIGH, and FGCHGLOW Registers. The charge and discharge counters are not affected by the latching operation.

### ADC Conversion Start and Stop

Setting the FGMUX bits D4 and D5 to '01' will cause the Fuel Gauge charge and discharge counter values to be latched at the beginning and end of each ADC conversion. This method of latching the Fuel Gauge counters allows the simultaneous measurement of cell voltage and current, which can be used to determine instantaneous cell impedance.

### TIMERA Overflow

The Fuel Gauge charge and discharge counter values can be latched by each TIMERA overflow by setting the FGMUX bits D4 and D5 to '10'. TIMERA is a 16-bit programmable counter that can be clocked by several sources. This method for latching the Fuel Gauge counters is for determining the accumulated charge/discharge for an arbitrary time interval.

### TIMERB Overflow

The Fuel Gauge charge and discharge counter values can be latched by each TIMERB overflow by setting the FGMUX bits D4 and D5 to '11'. TIMERB is clocked by the 32KHz Oscillator and is generally programmed to overflow at 1sec intervals. This latching method can then be used to capture charge and discharge counter values accumulated between the 1sec overflows. The difference between two successive counter values is the amount of charge that flows in 1sec, which is also the instantaneous current that is flowing.

### Arbitration Logic

The charge and discharge counters increment asynchronous with respect to the MAX1780 CPU instruction execution. Also, each of the four counter latching sources can occur asynchronously with respect to charge and discharge counter updates. To insure that erroneous counter values are not latched, the Fuel Gauge employs special arbitration logic. There is no arbitration while switching modes; therefore it is prudent to only trust data that has been latched by the most recently selected mode.

## Fuel Gauge Register Descriptions

### FGCONFIG1 (Read/Write) PORTB Address = 0x18

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | FGON | 0 | Writing a "1" turns ON the Fuel Gauge Block |
| D6 | FGCALON | 0 | 0 = Track the charge flow between CS+ and CS-<br>1 = Internally connects CS+ to CS-. Note the CS+ pin is high impedance. |
| D5<br>D4 | FGMUX[1:0] | 00 | FGMUX[1:0] chooses the source signal for latching counter data.<br>00 = Enables functions in FGCONFIG2<br>01 = Charge and discharge counts are latched at the start and completion of an ADC conversion.<br>10 = Charge and discharge counts are latched by each TIMERA overflow.<br>11 = Charge and discharge counts are latched by each TIMERB overflow. |
| D3 | FGCHGSTAT | - | 0 = Discharge current direction.<br>1 = Charge current direction.<br>(This bit is READ ONLY) |
| D2 | RFU | 0 | Read Only |
| D1 | RFU | 0 | Read Only |
| D0 | RFU | 0 | Read Only |

### FGCONFIG2 (Write Only) PORTB Address = 0x19

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | FGLATCHNOW | - | * Writing a "1" into this bit latches the contents of the Fuel Gauge counters into FGDISHIGH, FGDISLOW, FGCHGHIGH, and FGCHGLOW. |
| D6 | FGCLRDIS | - | * Writing a '1' into this bit clears the discharge counter |
| D5 | FGCLRCHG | - | * Writing a '1' into this bit clears the charge counter |
| D4 | RFU | | |
| D3 | RFU | | |
| D2 | RFU | | |
| D1 | RFU | | |
| D0 | RFU | | |

* Note: This Register is only operational when FGMUX[1:0] = 00

### FGDISLOW Register (Read Only): PORTB Address = 0x1A

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | FGD_D7 | 0 | Fuel Gauge discharge count latch Bit 7 |
| D6 | FGD_D6 | 0 | Fuel Gauge discharge count latch Bit 6 |
| D5 | FGD_D5 | 0 | Fuel Gauge discharge count latch Bit 5 |
| D4 | FGD_D4 | 0 | Fuel Gauge discharge count latch Bit 4 |
| D3 | FGD_D3 | 0 | Fuel Gauge discharge count latch Bit 3 |
| D2 | FGD_D2 | 0 | Fuel Gauge discharge count latch Bit 2 |
| D1 | FGD_D1 | 0 | Fuel Gauge discharge count latch Bit 1 |
| D0 | FGD_D0 | 0 | Fuel Gauge discharge count latch Bit 0 |

### FGDISHIGH Register (Read Only): PORTB Address = 0x1B

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | FGD_D15 | 0 | Fuel Gauge discharge count latch Bit 15 |
| D6 | FGD_D14 | 0 | Fuel Gauge discharge count latch Bit 14 |
| D5 | FGD_D13 | 0 | Fuel Gauge discharge count latch Bit 13 |
| D4 | FGD_D12 | 0 | Fuel Gauge discharge count latch Bit 12 |
| D3 | FGD_D11 | 0 | Fuel Gauge discharge count latch Bit 11 |
| D2 | FGD_D10 | 0 | Fuel Gauge discharge count latch Bit 10 |
| D1 | FGD_D9 | 0 | Fuel Gauge discharge count latch Bit 9 |
| D0 | FGD_D8 | 0 | Fuel Gauge discharge count latch Bit 8 |

### FGCHGLOW Register (Read Only): PORTB Address = 0x1C

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | FCD_D7 | 0 | Fuel Gauge charge count latch Bit 7 |
| D6 | FCD_D6 | 0 | Fuel Gauge charge count latch Bit 6 |
| D5 | FCD_D5 | 0 | Fuel Gauge charge count latch Bit 5 |
| D4 | FCD_D4 | 0 | Fuel Gauge charge count latch Bit 4 |
| D3 | FCD_D3 | 0 | Fuel Gauge charge count latch Bit 3 |
| D2 | FCD_D2 | 0 | Fuel Gauge charge count latch Bit 2 |
| D1 | FCD_D1 | 0 | Fuel Gauge charge count latch Bit 1 |
| D0 | FCD_D0 | 0 | Fuel Gauge charge count latch Bit 0 |

### FGCHGHIGH Register (Read Only): PORTB Address = 0x1D

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | FGC_D15 | 0 | Fuel Gauge charge count latch Bit 15 |
| D6 | FGC_D14 | 0 | Fuel Gauge charge count latch Bit 14 |
| D5 | FGC_D13 | 0 | Fuel Gauge charge count latch Bit 13 |
| D4 | FGC_D12 | 0 | Fuel Gauge charge count latch Bit 12 |
| D3 | FGC_D11 | 0 | Fuel Gauge charge count latch Bit 11 |
| D2 | FGC_D10 | 0 | Fuel Gauge charge count latch Bit 10 |
| D1 | FGC_D9 | 0 | Fuel Gauge charge count latch Bit 9 |
| D0 | FGC_D8 | 0 | Fuel Gauge charge count latch Bit 8 |

# Data Acquisition Unit

## General Description

The MAX1780 Data Acquisition Unit combines a high voltage Analog Front End/Multiplexer (AFE), and precision integrating Analog To Digital Converter (ADC). The AFE can be directly connected with up to four series lithium ion cells, and can be configured to provide eighteen different voltage measurements. The Analog-To-Digital Converter can be configured to make conversions at four resolutions: 11, 13, 15, and 16 bits, and will automatically shutdown after conversions to conserve power. Communication to and from the Data Acquisition Unit is via the PORTB peripheral interface. Figure 16 below shows the architecture of the unit.

## Features

- Differential Measurement Of Individual Cell Voltages
- Automatic Cancellation Of AFE Input Bias Currents
- User Selectable ADC Resolutions (11, 13, 15 and 16 bits)
- Automatic Shutdown Upon Conversion Completion
- Conversions Independent Of CPU Operation
- User maskable Interrupt upon end of conversion
- On-chip Temperature Sensor
- Maximum Differential Cell Voltage Measurement Error +/-50mV

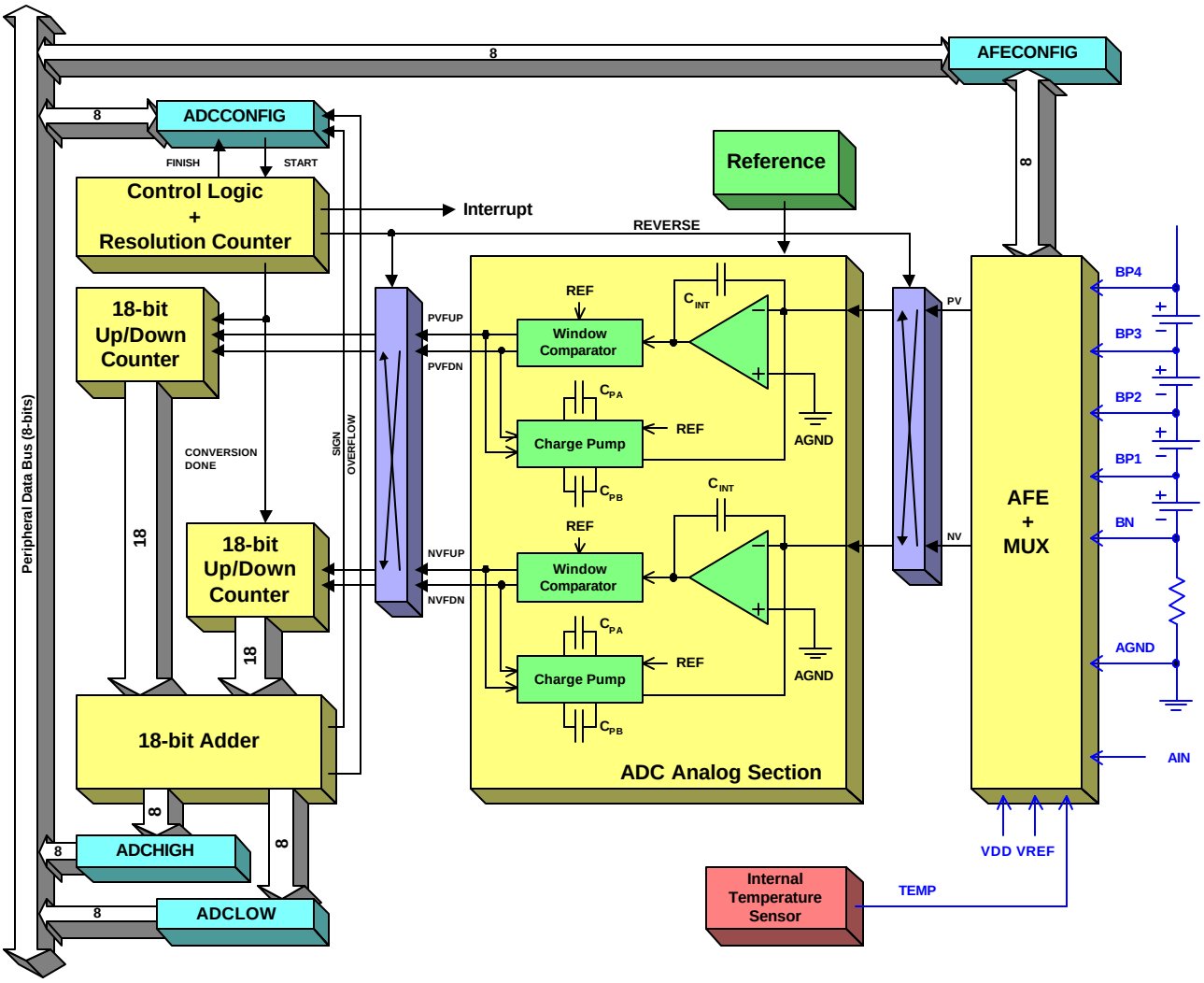



**Figure 16, Data Acquisition Unit Block Diagram**

## Analog Front End/Multiplexer (AFE)

The Analog Front End selects and scales the desired input signal for the ADC. The AFE allows the user to make four types of precision measurements selectable through the AFEMODE bits in the AFECONFIG Register:

| Voltage Range | Connection Type |
|---|---|
| 0 to 5.12V | Single-ended referred to AGND |
| 0 to 20.48V | Single-ended referred to BN |
| 0 to 20.48V | Single-ended referred to AGND |
| 0 to 20.48V | Differential |

The AFESELECT bits in the AFECONFIG register allow the user to choose a particular signal within each Mode. This equates to a total of 14 Single-ended and 4 Differential measurements.

### Input Bias Cancellation

While the MAX1780's ADC measures battery pack cell voltages, each connection to the ADC connects a 4 MegOhm resistor from the measured pin to ground during conversion. Although the resulting input bias current

is small, even small currents from intermediate cell stack voltages can imbalance a cell stack over time. To protect against this, the MAX1780 incorporates a special input bias current cancellation circuit (ICAN). Programming the number of series cells in the AFECONFIG register automatically enables the ICAN circuit for an intermediate cell voltage measurement during a conversion. Note that while the ICAN circuit is enabled, about 65uA of additional supply current is drawn from the B4P pin for each pin whose input bias current is being cancelled. When enabled, the ICAN circuit will typically reduce the uncanceled input bias current by a factor of 100.

## AFE Register Descriptions

### AFECONFIG (Read/Write) PORTB Address 0x10

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | - | 0 | Unused R/W bit. |
| D6-D5 | CELLCNT | 0 | Selection of battery pack cell count: (Used only for input bias current cancellation)<br>00 = Input Bias Current Cancellation Off.<br>01 = 2 Cells. Input bias current canceled for B1P only.<br>10 = 3 Cells. Input bias current canceled for B1P and B2P.<br>11 = 4 Cells. Input bias current canceled for B1P, B2P, and B3P.<br>Note: Refer to Figure 27 for the proper connections for 2, 3, and 4 series cell configurations. |
| D4-D3 | AFEMODE | 00 | Selection of ADC measurement Mode:<br>00 = Configures AFE for low-voltage measurements, referred to AGND<br>01 = Configures AFE for high-voltage measurements, referred to BN<br>10 = Configures AFE for high-voltage measurements, referred to AGND<br>11 = Configures AFE for high-voltage differential measurements |
| D2-D0 | AFESELECT | 000 | Selection of analog signal to measure:<br><br>AFEMODE = 00 (Low-voltage, Range: 0 to 5.12V):<br>000 = AGND, Referred to AGND<br>001 = Internal Temperature Measurement, Referred to AGND<br>010 = VDD, Referred to AGND<br>011 = VREF, Referred to AGND<br>100 = BN, Referred to AGND<br>101 = AIN, Referred to AGND<br>11x = undefined<br><br>AFEMODE = 01 (High-voltage, Range: 0 to 20.48V):<br>000 = B1P, Referred to BN<br>001 = B2P, Referred to BN<br>010 = B3P, Referred to BN<br>011 = B4P, Referred to BN<br>1xx = undefined<br><br>AFEMODE = 10 (High-voltage, Range: 0 to 20.48V):<br>000 = B1P, Referred to AGND<br>001 = B2P, Referred to AGND<br>010 = B3P, Referred to AGND<br>011 = B4P, Referred to AGND<br>1xx = undefined |

| | | | |
|---|---|---|---|
| | | | AFEMODE = 11 (High-voltage differential, Range: 0 to 20.48V):<br>000 = B1P – BN<br>001 = B2P – B1P<br>010 = B3P – B2P<br>011 = B4P – B3P<br>1xx = undefined |

# Analog-To-Digital Converter (ADC)

## Operation

To perform an ADC conversion, first program the AFECONFIG Register with the number of series cells, and then select the desired measurement mode. Start the conversion by setting the ADCSTCONV bit to a "1" in the ADCCONFIG register. This will cause the peripheral to power-up and begin a conversion. Upon completion of the conversion, the control logic clears the ADCSTCONV bit in the ADCCONFIG register. The conversion result can be read from the ADCHIGH and ADCLOW Registers. If the ADCINTON bit in the ADCCONFIG register is set to "1", an interrupt will occur at the beginning and end of each ADC conversion. At the end of each conversion cycle the ADC will automatically power itself down to conserve energy.

## Dual Voltage-To-Frequency Converter

Each V-to-F Converter changes an applied input voltage to a proportional output frequency. This frequency is in-turn sent to a window comparator that determines if an LSB of change with respect to the reference has occurred. PVFUP and NVFUP pulses are generated for each LSB change in the positive direction, and correspondingly PVFDN and NVFDN pulses are generated for each LSB change in the negative direction. Each V-to-F converter has a charge pump which restores current to the summing node of the converter.

## Digital Counter/Adder

There are two 18-bit up/down counters whose counts are 1's complement added to obtain a 16-bit binary conversion value. Although the highest resolution conversion is 16 bits, the counters and adder are 18 bits wide to implement sign and overflow. The final ADC result is formed by an 18-bit adder, which sums the two counter outputs. The MAX1780 CPU can read the conversion results as two bytes.

## Control Logic And Resolution Counter Block

The Control Logic And Resolution Counter Block is clocked by the 32KHz oscillator, and interprets the ADCCONFIG Register bits. It controls the state timing of a conversion cycle and as well as determining the conversion resolution.

## Over-Range Status And Limit Bits

The MAX1780 Data Acquisition Unit has special circuitry to handle and report status on conversions where the measured input voltage is outside of the ADC's full-scale voltage ranges of 5.12V or 20.48V. The SIGN and OVERFLOW bits provide over-range status information, and the LIMIT bit restricts ADC conversion results.

### The OVERFLOW Bit

Whenever an ADC conversion is attempted on a positive or negative voltage outside the selected range, the OVERFLOW bit, D7 in the ADCCONFIG Register, will be set.

### The SIGN Bit

The SIGN Bit, D6 in ADCCONFIG Register, will be set whenever an ADC conversion results in a negative value.

### The LIMIT Bit

The Limiting function, when enabled, will limit positive over-range conversion results to 0xFFFF, and negative over-range values to 0x0000. When Limiting is disabled, over-range conversion results will roll over. To enable the Limiting function, set bit D5 in the ADCCONFIG Register to '1'.

## Understanding ADC Error Sources

This ADC has four sources of error:

**Gain Error:** The ADC gain error is influenced by trimming resolution, temperature coefficient of the precision reference and v-to-f converters, gain matching between the various input channels of the ADC, and accuracy of the 32768Hz crystal time base. Note that it may take up to 4 seconds for the 32KHz oscillator to stabilize after power up. The ADC is trimmed to be most accurate at the B4P to B1P inputs. This error will total to about 0.5%.

**Offset Error:** The input offset of the two v-to-f converters will introduce an offset error into the result. The typical offset error is around 100uV and becomes negligible in most conversions.

**Common-Mode Error:** This source of error only comes into play for high-voltage differential measurements. Matching between the positive and negative input channels of the measurement will introduce an error proportional to the common-mode voltage. What this means is if B4P=B3P=20V, the converter may measure 10mV; when B4P=B3P=10V, this same converter would measure 5mV.

**Quantization Error:** Each v-to-f converter has +/-0.5 bits of quantization error. When the results of the two v-to-f converters are digitally subtracted, this becomes +/-1 bits of quantization error. Because each full conversion actually consists of two consecutive conversions with the v-to-f converters swapped, the total quantization error becomes +/-2 bits. Note that although it is theoretically possible to see 2 bits of quantization error, it is unlikely, and most conversions will show +/-0.5 bits of quantization error. Still, for the various resolutions, the worse case quantization error becomes:

| Resolution | Effective Resolution | HV quantization error | LV quantization error |
|---|---|---|---|
| 11 bits | 9 bits | 40mV | 10mV |
| 13 bits | 11 bits | 10mV | 2.5mV |
| 15 bits | 13 bits | 2.5mV | 0.625mV |
| 16 bits | 14 bits | 1.25mV | 0.3125mV |

## Effective ADC Resolution

As explained in the "Understanding ADC Error Sources" section, each ADC conversion cycle averages the results of two sub-conversions with the V-to-F converters swapped. The digital subtraction in conjunction with the V-to-F converter swapping creates +/-2 LSB's of quantization error. Therefore, the effective ADC resolutions are as shown in Figure 17 below.

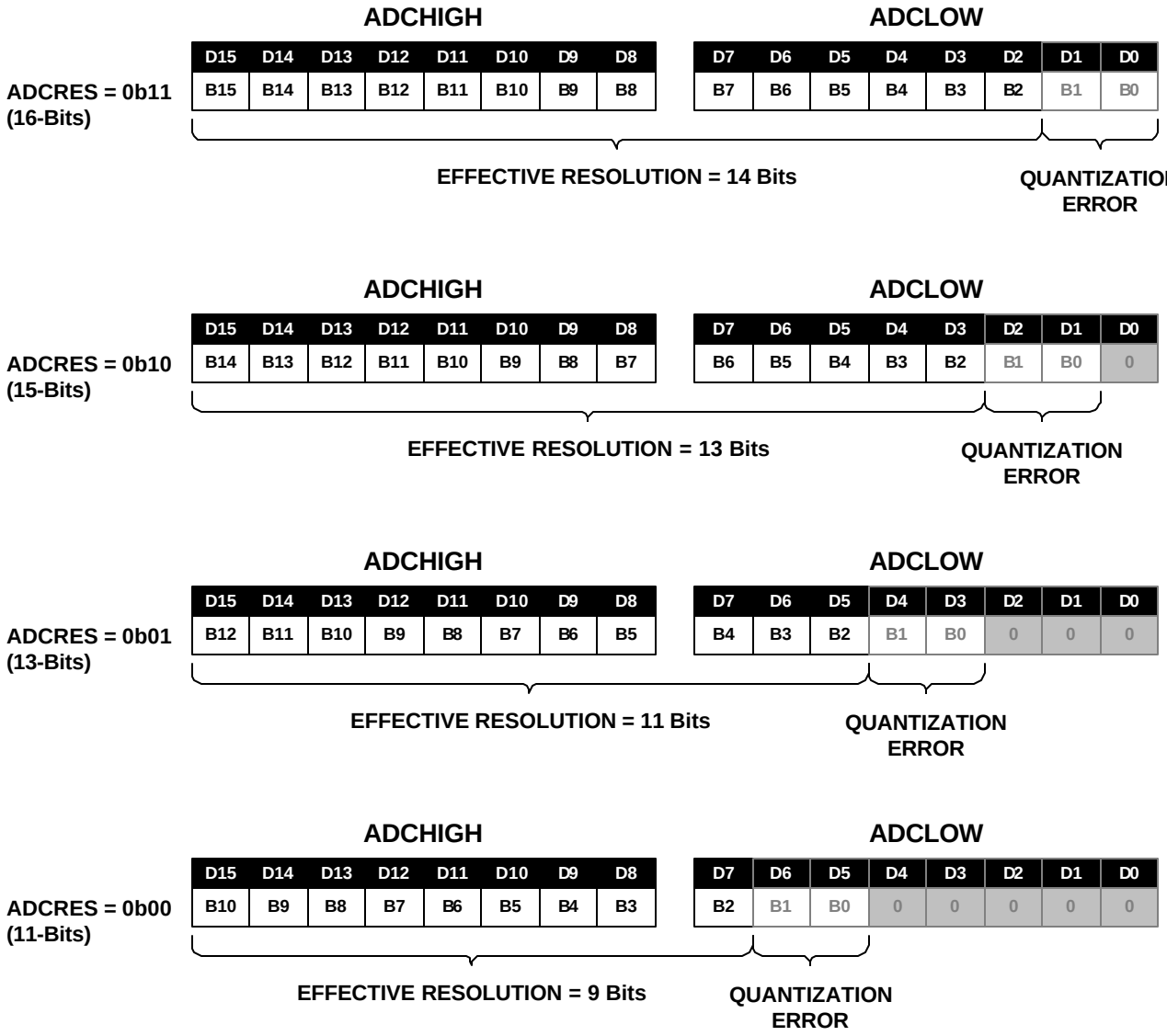



Figure 17, Effective ADC Resolutions

## ADC Register Descriptions

### ADCCONFIG (Read/Write) PORTB Address = 0x11

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | OVERFLOW | 0 | 0 = In Range<br>1 = Overflow |
| D6 | SIGN | 0 | 0 = Positive overflow<br>1 = Negative overflow |
| D5 | ADCLIMIT | 1 | 1=Limiting On.<br>0=Limiting Off.<br>Limiting only affects what is read from ADC_LO/ADC_HI = {0x12 and 0x13}. Conversion does not have to be repeated to toggle between limited and non-limited results. |
| D4 | CHOPON | 1 | 1 = Offset cancellation enabled. (Always leave on for best performance)<br>0 = Offset cancellation disabled. |
| D3 | ADCINTON | 0 | 1= Enables ADC interrupts. Interrupts occur at the beginning and end of a conversion cycles. |
| D2<br>D1 | ADCRES[1:0] | 01 | 00: Resolution 11 bits. (9 bits effective with +/-2 bits of quantization error)<br>01: Resolution 13 bits. (11 bits effective with +/-2 bits of quantization error)<br>10: Resolution 15 bits. (13 bits effective with +/-2 bits of quantization error)<br>11: Resolution 16 bits. (14 bits effective with +/-2 bits of quantization error) |
| D0 | ADCSTCONV | 0 | Writing a “1” to this bit starts a conversion cycle. Upon completion of a conversion cycle, the ADC logic will reset this bit to “0” indicating that sampled data is ready to be read. |

### ADCLOW (Read Only) PORTB Address = 0x12

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | ADC_D7 | 0 | ADC Sample Bit 7. |
| D6 | ADC_D6 | 0 | ADC Sample Bit 6 |
| D5 | ADC_D5 | 0 | ADC Sample Bit 5. (11 Bit LSB) |
| D4 | ADC_D4 | 0 | ADC Sample Bit 4. |
| D3 | ADC_D3 | 0 | ADC Sample Bit 3. (13 Bit LSB) |
| D2 | ADC_D2 | 0 | ADC Sample Bit 2. |
| D1 | ADC_D1 | 0 | ADC Sample Bit 1. (15 Bit LSB) |
| D0 | ADC_D0 | 0 | ADC Sample Bit 0. (16 Bit LSB) |

**ADCHIGH (Read Only)** **PORTB Address = 0x13**

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | ADC_D15 | 0 | ADC Sample Bit 15. (11/13/15/16 Bit MSB) |
| D6 | ADC_D14 | 0 | ADC Sample Bit 14. |
| D5 | ADC_D13 | 0 | ADC Sample Bit 13. |
| D4 | ADC_D12 | 0 | ADC Sample Bit 12. |
| D3 | ADC_D11 | 0 | ADC Sample Bit 11. |
| D2 | ADC_D10 | 0 | ADC Sample Bit 10. |
| D1 | ADC_D9 | 0 | ADC Sample Bit 9. |
| D0 | ADC_D8 | 0 | ADC Sample Bit 8. |

# Temperature Sensor

## Description

The MAX1780 has an on-chip temperature sensor, a bi-polar Proportional-To-Absolute-Temperature (PTAT) transistor, which closely tracks the MAX1780 die temperature. The temperature measurement results are in degrees Kelvin.

## Operation

To make a temperature measurement, use the following procedure:

1. Set the AFECONFIG Register AFEMODE[4:3] bits to '00'.
2. Set the AFECONFIG Register AFESELECT[2:0] bits to '001' to select the on-chip temperature sensor.
3. Write a '1' to the ADCCONFIG Register ADCSTCONV bit to trigger an ADC conversion.
4. Read the measurement results from the ADCHIGH and ADCLOW Registers.

Figure 18 below depicts the individual bit weights, in degrees Kelvin, for temperature measurement conversion values at 16 bits of resolution. An 11-bit conversion will have a worse case quantization error of +/- 1°K. To convert the temperature in degrees Kelvin to degrees Celsius, subtract 0x8893 from the measured result.

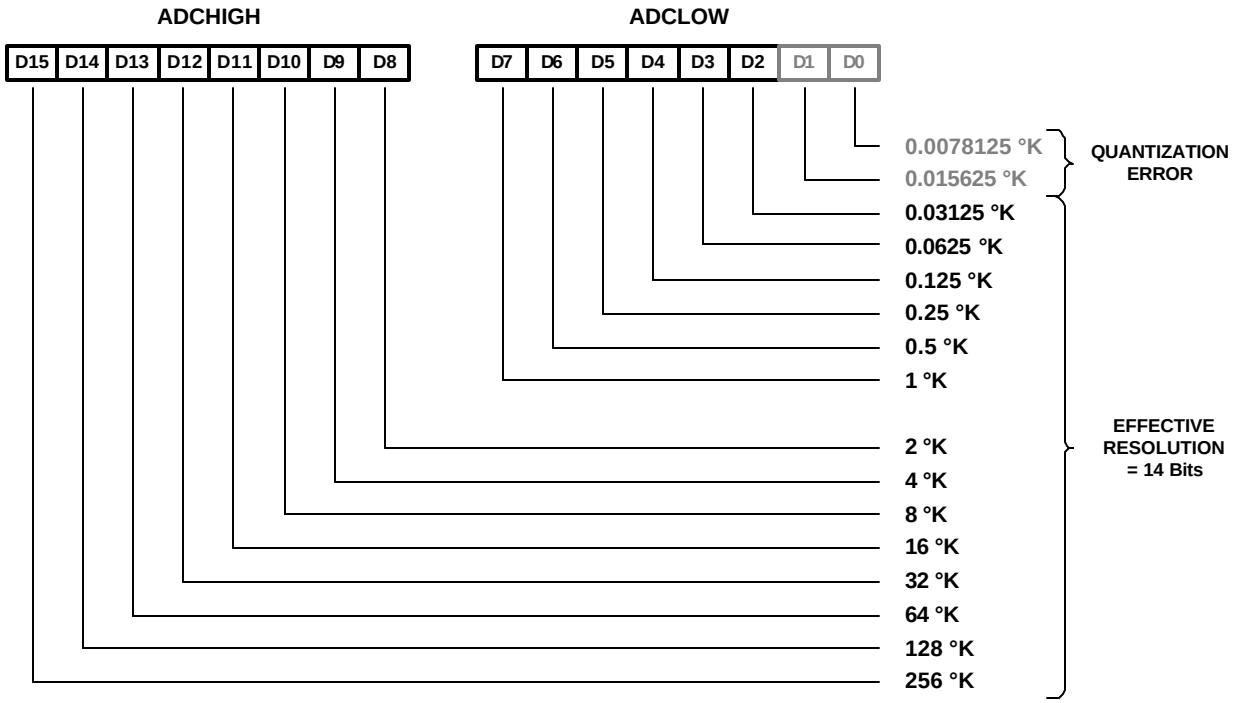



Figure 18, Temperature Conversion Bit Weights

# Overcurrent Protection Block

## Description

The MAX1780's Overcurrent Protection Block continuously monitors the current flowing through the battery pack sense resistor and compares this with the user adjustable thresholds for Overcharge and Overdischarge current. The Overcurrent Protection block will continue to provide protection even when the MAX1780 processor core is shutdown (SLEEP). Figure 19 below shows the functional diagram of the over-current comparator section. An overcurrent condition occurs whenever the voltage on CS+ exceeds the voltage on OCI (for charging currents), or when ODI falls below CS- (for discharging currents). When an over-current condition occurs, the overcurrent comparators generate an interrupt to the processor core, as well as set the OD (discharging) or OC (charging) latch. These latches remain set until a '1' is written to the appropriate bit (D2 or D3) of the INTRSTAT Register, or the MAX1780 initiates a power-on reset. A logic block follows the latch, which sets the gate-driver output's appropriate state, as defined in Table 1 and Table 2, and drives the N-channel MOSFET open-drain gate drivers.

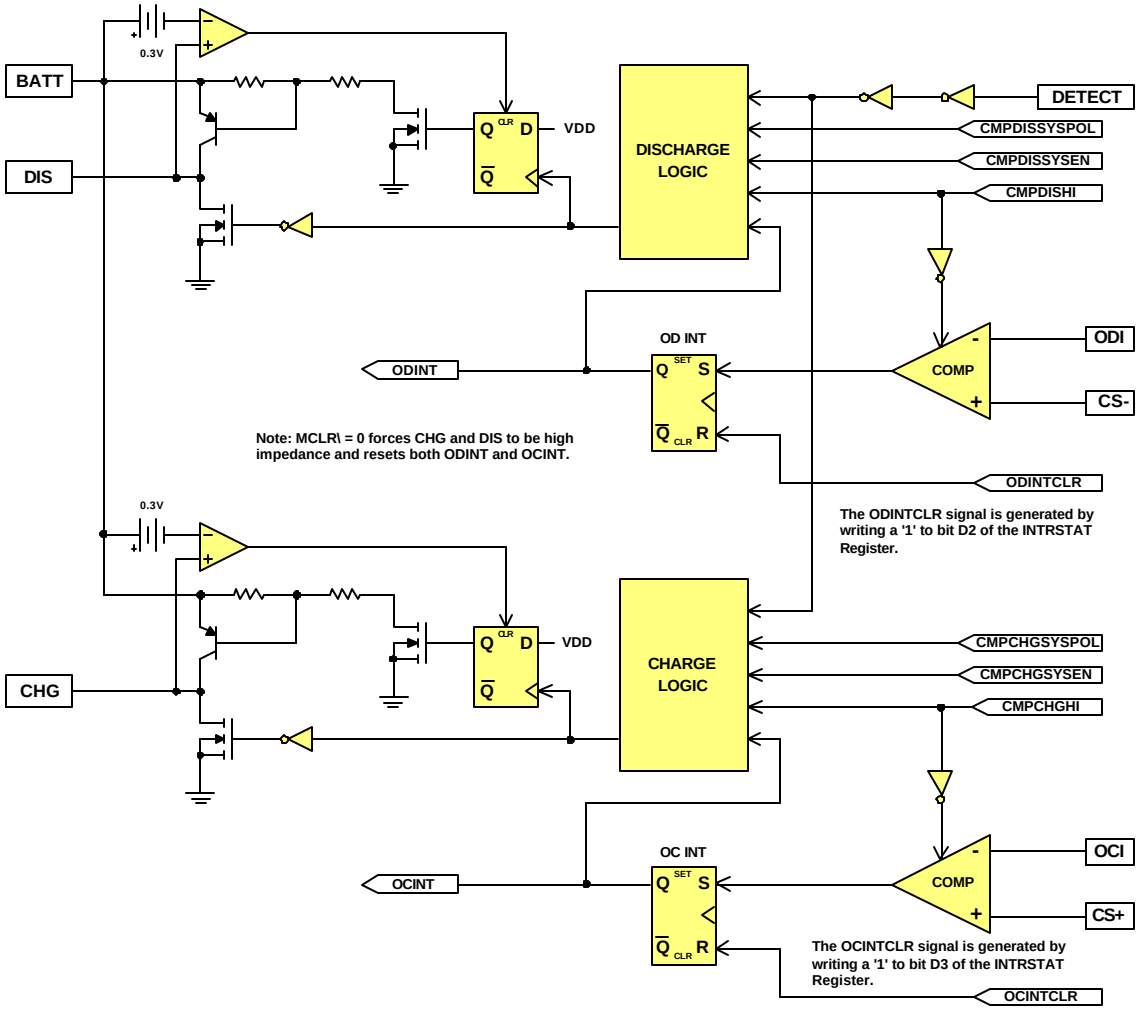



**Figure 19, Overcurrent Comparator Functional Diagram**

### DISCHARGE LOGIC

| $\overline{MCLR}$ | ODINT | CMPDISHI | CMPDISSYSEN | CMPDISSYSPOL | DETECT | DIS |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | X | X | Asserted Low |
| 1 | 0 | 0 | 1 | 0 | 0 | Asserted Low |
| 1 | 0 | 0 | 1 | 0 | 1 | Released High |
| 1 | 0 | 0 | 1 | 1 | 0 | Released High |
| 1 | 0 | 0 | 1 | 1 | 1 | Asserted Low |
| 1 | 0 | 1 | X | X | X | Released High |
| 1 | 1 | X | X | X | X | Released High |
| 0 | X | X | X | X | X | Released High |

**Table 1, Overdischarge Logic Truth Table**

### CHARGE LOGIC

| $\overline{MCLR}$ | OCINT | CMPCHGHI | CMPCHGSYSEN | CMPCHGSYSPOL | DETECT | CHG |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | X | X | Asserted Low |
| 1 | 0 | 0 | 1 | 0 | 0 | Asserted Low |
| 1 | 0 | 0 | 1 | 0 | 1 | Released High |
| 1 | 0 | 0 | 1 | 1 | 0 | Released High |
| 1 | 0 | 0 | 1 | 1 | 1 | Asserted Low |
| 1 | 0 | 1 | X | X | X | Released High |
| 1 | 1 | X | X | X | X | Released High |
| 0 | X | X | X | X | X | Released High |

**Table 2, Overcharge Logic Truth Table**

## Using Software To Control The Protection MOSFETs

Although control of the Charge and Discharge Overcurrent MOSFETs is handled by the Overcurrent Protection Block, they can also be switched ON and OFF under software control. This is accomplished by writing to bits D0 and D3 of the CMPREG Register. Writing a '1' to either bit turns the respective protection MOSFET OFF, and writing a '0' turns it ON. Users should be aware that turning ON either of the protection MOSFETs under software control, can possibility cause a spurious ODI/OCI interrupt. Please use the following procedure when controlling the protection MOSFETs with software:

1. Mask ODI/OCI interrupts.
2. Turn ON the desired protection MOSFET.
3. Clear ODI and OCI interrupt flags in INTRSTAT Register.
4. Re-enable ODI/OCI interrupts.

## Clearing Overcurrent Interrupts

Whenever the Overcurrent Protection Block turns OFF either the Charge or Discharge protection MOSFET, an interrupt is generated. Then the corresponding interrupt status bit (OCINT or ODINT) will be set in the INTRSTAT Register of the Interrupt Control Block. After each ODINT/OCINT event, the user should clear the respective interrupt status bit by writing a '1' to either bit D2 or bit D3 of the INTRSTAT Register.

**CMPREG Register (Write Only ): PORTB Address = 0x0A**

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | Unused | - | |
| D6 | Unused | - | |
| D5 | CMPCHGSYSEN | 0 | See Tables 1 and 2 for functionality. |
| D4 | CMPCHGSYSPOL | 0 | See Tables 1 and 2 for functionality. |
| D3 | CMPCHGHI | 0 | 0 – Turns ON the Charge MOSFET and overcharge current comparator.<br>1 – Turns OFF the Charge MOSFET and overcharge current comparator. |
| D2 | CMPDISSYSEN | 0 | See Tables 1 and 2 for functionality. |
| D1 | CMPDISSYSPOL | 0 | See Tables 1 and 2 for functionality. |
| D0 | CMPDISHI | 0 | 0 – Turns ON the Discharge MOSFET and overdischarge current comparator.<br>1 – Turns OFF the Discharge MOSFET and overdischarge current comparator. |

# High-Voltage Output Port

## Description

The High-Voltage Output Block gives the user the ability to switch high-voltage (up to BATT) nodes. Six outputs HV0 – HV5, can be asserted LOW to pull-down nodes normally tied HIGH (pulled-up to BATT). Two outputs HV6, and HV7, can be asserted HIGH to pull-up nodes normally tied LOW (pulled-down to AGND). All eight outputs are protected against ESD. The HV0 – HV5 pins are normally used to switch the battery pack capacity indicator LEDs ON and OFF. They could also be used to drive the external P-Channel MOSFETs of a charge balancing circuit.

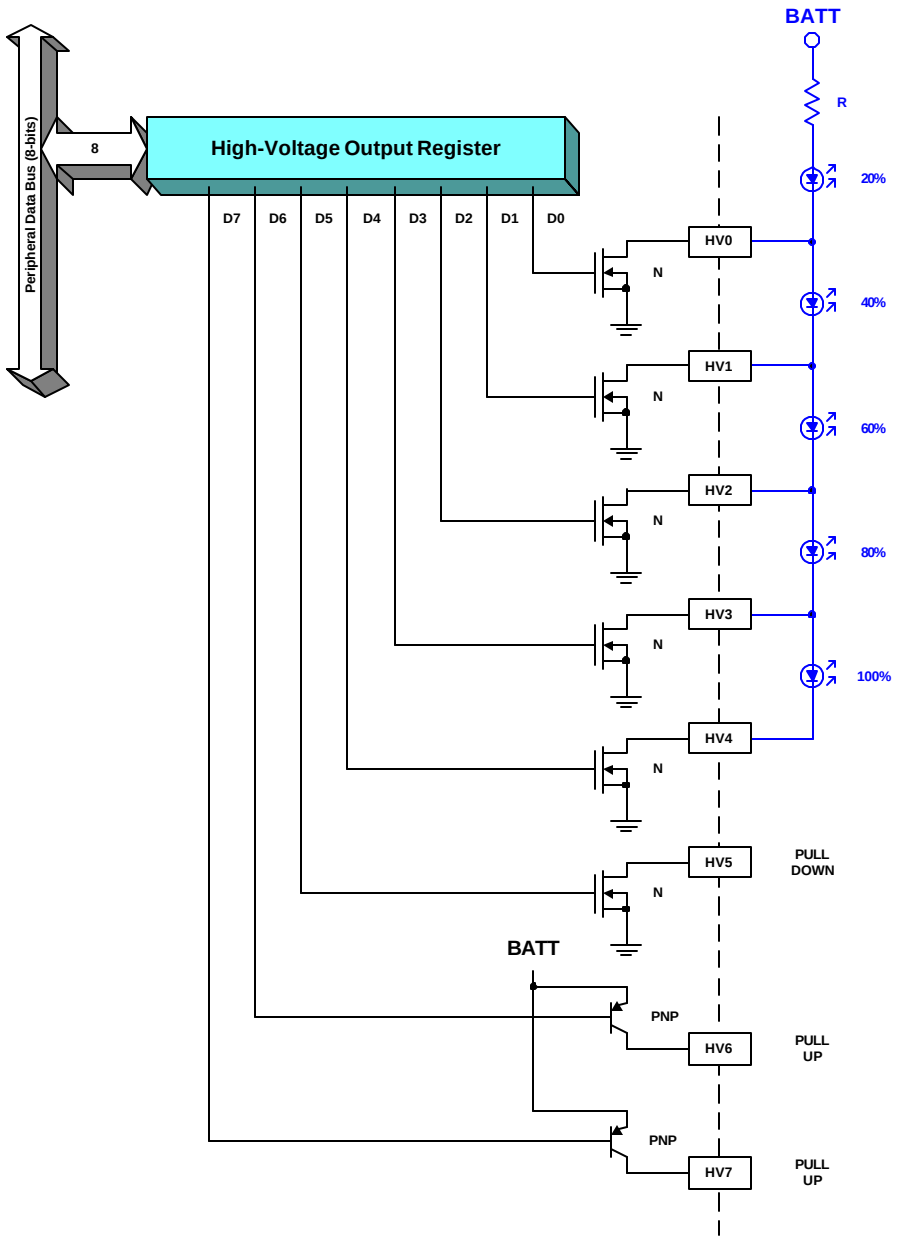



**Figure 20, High-Voltage Output Port**

## Operation

Writing an 8-bit value to the block's HVO Register sets or clears the corresponding bit (HV0 – HV7), this in turn asserts or releases the corresponding High-Voltage Output pin (HV0 – HV7) of the port.

## High-Voltage Output Port Register Description

**HVO (Read/Write)** **PORTB Address = 0x0B**

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | HV7 | 0 | 0 – Pin HV7 is high impedance.<br>1 – Pin HV7 is pulled up to BATT. |
| D6 | HV6 | 0 | 0 – Pin HV6 is high impedance.<br>1 – Pin HV6 is pulled up to BATT. |
| D5 | HV5 | 0 | 0 – Pin HV5 is high impedance.<br>1 – Pin HV5 is pulled to GND. |
| D4 | HV4 | 0 | 0 – Pin HV4 is high impedance.<br>1 – Pin HV4 is pulled to GND. |
| D3 | HV3 | 0 | 0 – Pin HV3 is high impedance.<br>1 – Pin HV3 is pulled to GND. |
| D2 | HV2 | 0 | 0 – Pin HV2 is high impedance.<br>1 – Pin HV2 is pulled to GND. |
| D1 | HV1 | 0 | 0 – Pin HV1 is high impedance.<br>1 – Pin HV1 is pulled to GND. |
| D0 | HV0 | 0 | 0 – Pin HV0 is high impedance.<br>1 – Pin HV0 is pulled to GND. |

# Digital Peripherals

## High-Speed SPI Interface

### Description

The MAX1780 uses the SPI Interface to read and write to an external SPI EEPROM. It can communicate with the latest technology 5MHZ EEPROMs using “turbo mode”, or interface to the older generation 1MHz EEPROMs. With this architecture, the MAX1780 CPU can load program instructions/data from the serial EEPROM into its instruction RAM for execution. Once the block is configured, data is transferred through the SPI port by reading and writing to the SPIDATA register. Generation of the clock (SCLK) and data out (SO) signals is automatic, performed by the MAX1780 hardware.

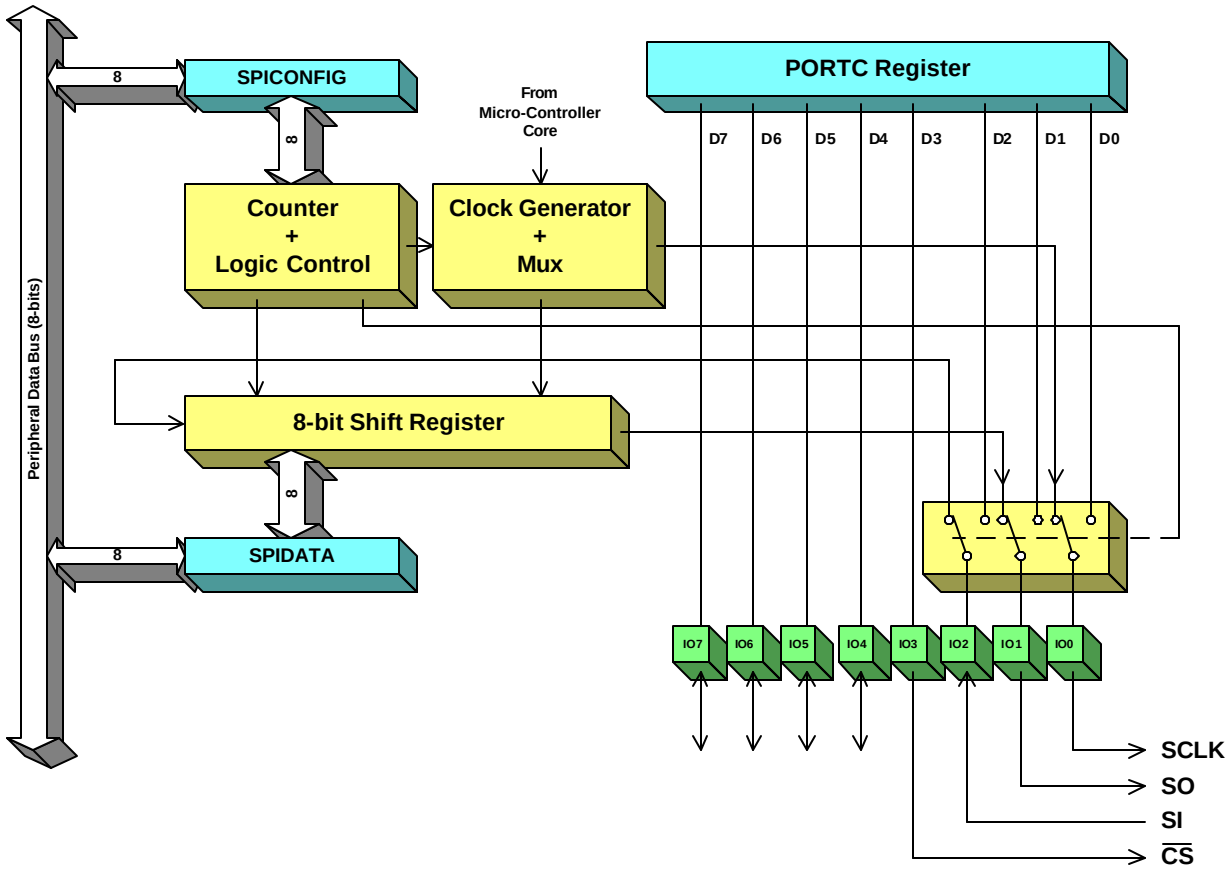



**Figure 21, High-Speed SPI Port**

### Operation

The SPICONFIG register needs to be set prior to any SPI operation. The POR value of this register is designed to allow normal operation of the MAX1780 IO pins IO2/SI, IO1/SO, and IO0/SCLK. The SPIENABLE bit of SPICONFIG must be set to a 1 for SPI operations. The speed needs to be selected via the SPISPEED bit, with 1 for Turbo Mode and 0 for Normal Mode.

The MAX1780 SPI Interface is a Master, performing simultaneous read and write operations. As data is written to the IO1/SO pin, data is read into the IO2/SI pin. The data to be fed out serially the IO1/SO pin is written to the SPIDATA register. The input data returned via the serial port may be read from the same SPIDATA register.

Programs retrieving SPI data after a data transfer must wait 3 instruction cycles when the interface is in Turbo Mode, and 6 instruction cycles for Normal Mode. During this period, the MAX1780 CPU is free to execute other instructions.

Because the SPI Interface shares use with the Port C IO[2:0] pins, the voltage levels are defined in the Logic Inputs/Outputs section of the Electrical Characteristics Table. Note that the SPI Chip Select timing is controlled using normal IO operation of pin IO3. User programs must enable/disable the SPI Chip Select as required.

## SPI Interface Register Descriptions

### SPIDATA Register (Read/Write): PORTB Address = 0x0C

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | | - | Data bit 7 |
| D6 | | - | Data bit 6 |
| D5 | | - | Data bit 5 |
| D4 | | - | Data bit 4 |
| D3 | | - | Data bit 3 |
| D2 | | - | Data bit 2 |
| D2 | | - | Data bit 1 |
| D2 | | - | Data bit 0 |

Note: Writing to this register causes data to be transferred on the SPI port and clocked through the shift register. Reading from this register causes no external events.

### SPICONFIG (Read/Write ): PORTB Address = 0x0D

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | | 0 | RFU (Always returns 0 if read) |
| D6 | | 0 | RFU (Always returns 0 if read) |
| D5 | | 0 | RFU (Always returns 0 if read) |
| D4 | BTEST | 0 | 0 – Normal Operation<br>1 – Factory Test Mode Only. |
| D3 | | 0 | RFU (Always returns 0 if read) |
| D2 | | 0 | RFU (Always returns 0 if read) |
| D1 | SPISPEED | 0 | 0 – Normal Mode (SCLK speed is INSTOSC/2)<br>1 – Turbo Mode (SCLK speed is INSTOSC). |
| D0 | SPIENABLE | 0 | 0 – pins IO0/SCLK, IO1/SO, and IO2/SI have standard PORTC IO functionality.<br>1 – pins IO0/SCLK, IO1/SO, and IO2/SI have SPI Interface functionality. |

# SMBus Interface

## Introduction

The System Management Bus (SMBus) is a two wire, bi-directional serial bus which provides a simple, efficient interface for data exchange between devices. The SMBus uses a serial data line (SDA) and a serial clock line (SCL) for data transfer. SMBus data is 8 bits in length. Data on the SMBus can be changed only when SCL is low and must be held stable when SCL is high. The MSB is transmitted first and each byte has to be followed by an acknowledge bit (ACK). An “ACKNOWLEDGE” is generated by the device receiving data by pulling the SDA line low on the 9th SCL clock cycle. Therefore one complete data byte transfer needs 9 SCL clocks. If the Master receiver does not acknowledge (NACK) the Slave transmitter after a byte has been transmitted, this signals an "end of data" to the Slave. The Slave will now release the SDA line allowing the Master to generate a "STOP" or "START" condition.

A complete specification for the SMBus can be found in the **System Management Bus Specification**, Revision 1.1, December 11, 1998.

## Features

- Master/Slave operation.
- Automatic SCL Hold.
- Two software programmable SMBus addresses.
- Four Master SCL clock frequencies.
- Completely Interrupt driven operation.
- Hardware Generation/Detection of SMBus START, Repeated START, and STOP conditions.
- SMBus Timeout Detector.
- Hardware Generation/Detection of the Acknowledge bit.
- Bus busy detection.

## Description

The MAX1780 SMBus Interface can be completely interrupt driven. This allows the MAX1780 processor core to either sleep or process other tasks while the relatively slow SMBus transactions are handled. The MAX1780 factory ROM code includes many subroutines that automate the operation of the SMBus Interface. For a detailed description of these routines consult the MAX1780 ROM Code Supplement.

The SMBus Interface is comprised of a transmitter and receiver and can function as both a Master and a Slave device. The Master transmitter can send SCL clocks at four user selectable speeds. The default Master SCL clock speed is 48.61 KHz. The Slave receiver can respond to two user programmable addresses at SCL clock speeds up to the full 100KHz specification. Once initialized, the Slave receiver will respond to SMBus communications to its address, even if the MAX1780 processor core is in SLEEP mode. Master mode operations require the MAX1780 instruction oscillator to be active to initiate START generation, byte operations, and STOP generation.

The SMBus Interface incorporates separate receive and transmit data shift registers, allowing it to operate in full duplex. This means that as it transmits on the SMBus, it receives the data being transmitted. This allows the SMBus control software to compare data sent with data received. Similarly, all of the commands generated by the transmitter; START, STOP, ACK, etc. are detected. This allows software frame checking of the complete SMBus transaction.

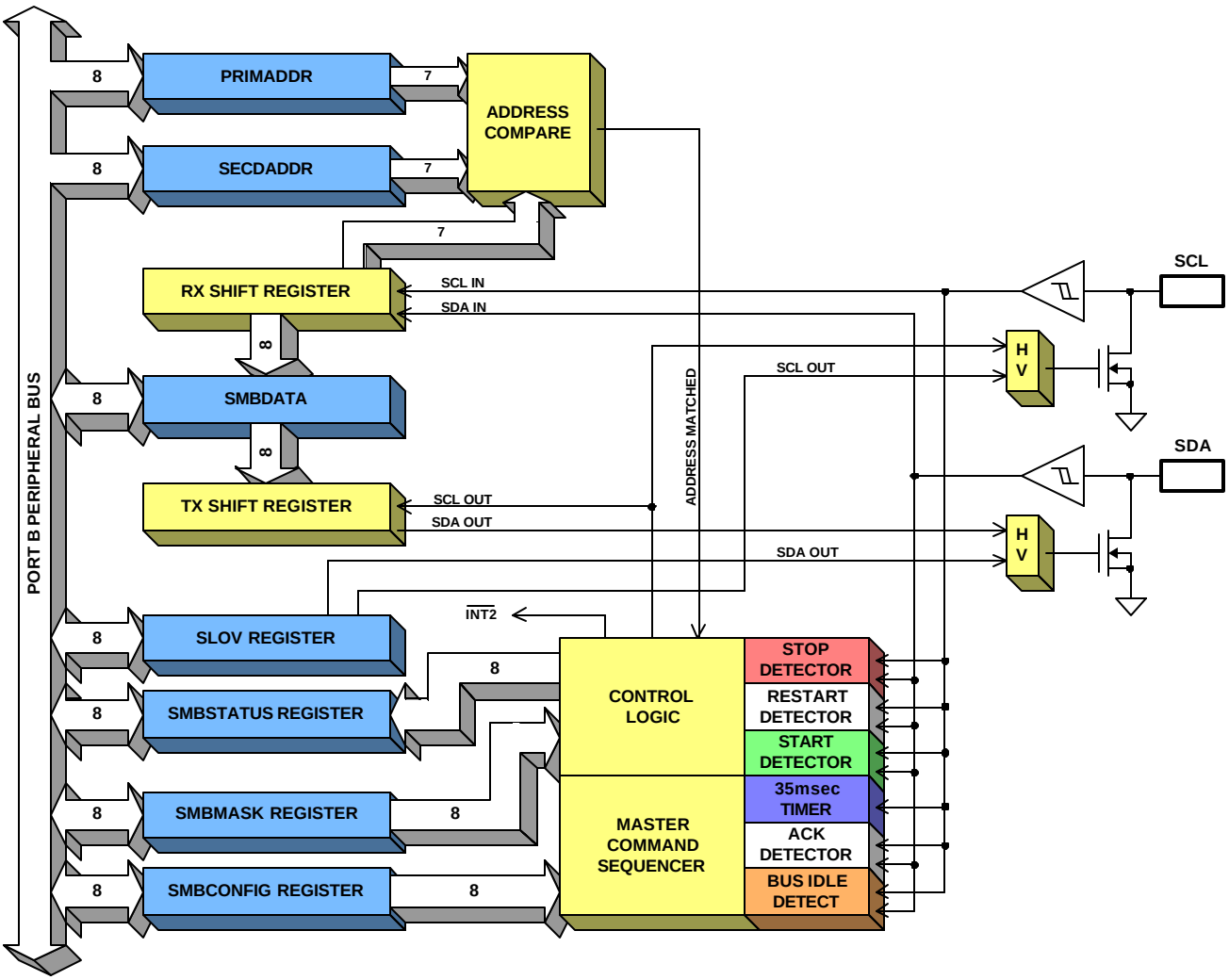



**Figure 22, SMBus Interface Block Diagram**

### Start Detector

The Start Detector will set the STARTDET flag in the SMBSTATUS Register whenever a valid START condition, (a falling edge on SDA while SCL is high) occurs on the SMBus. If the STMSK bit in the SMBMASK Register is set, an interrupt (INT2) will be generated.

### Restart Detector

The Restart Detector will set the RESTARTDET flag in the SMBSTATUS Register whenever a valid repeated Start condition (a falling edge on SDA while SCL is high and without a valid stop condition) occurs on the SMBus. If the REMSK bit in the SMBMASK Register is set, an interrupt (INT2) will be generated.

### Stop Detector

The Stop Detector will set the STOPDET flag in the SMBSTATUS Register whenever a valid STOP condition (a raising edge on SDA while SCL is high) occurs on the SMBus. If the SPMSK bit in the SMBMASK Register is set, an interrupt (INT2) will be generated.

### ACK Detector

The ACK Detector will set the ACKDET flag in the SMBSTATUS Register whenever a ACKNOWLEDGE (SDA line is low at the rising edge of the 9$^{th}$ SCL clock) occurs on the SMBus. If the ACKMSK bit in the SMBMASK Register is set, an interrupt (INT2) will be generated.

### ACK/NACK Generator

The ACK/NACK Generator's purpose is to assert the SDA line low after the falling edge of the 8th SCL clock and release the SDA line high after the falling edge of the 9th SCL clock. The ACK Generator is active in both Slave and Master Mode Receive operations, whenever the ACKNACK bit (D3) in the SMBCONFIG Register is reset to 0. There are two types of acknowledge pulses, Automatic and Conditional:

### Automatic ACK Generation After a Start or Restart Condition:

The first 7 bits of captured data will be compared to the values programmed into both SMBus device address registers. If there was a match with either of them, the ACK Generator will generate an acknowledge pulse by asserting the SDA line.

### Conditional ACK Generation In All Other Slave or Master Mode Receive Operations:

Reading the received data byte in the SMBDATA Latch releases the automatic SCL Hold.

### SCL Holding Detector

The SMBus Master uses the SCL Holding Detector to monitor each low to high transition of the SCL line for the possibility that the Slave is holding it low.

### Automatic SCL Hold Circuit

The purpose of the Automatic SCL Hold Circuit is to give the Slave a method for stretching the Master's SCL clock to allow it more time to get/give data being sent/requested by the Master. The Slave accomplishes this by asserting and holding the SCL line low after the Master transitions SCL from high to low after the falling edge of the 9th SCL clock following a Start or Restart condition. If the SCLHMSK bit is set, an interrupt will be generated to indicate the need to service the SMBDATA Register.

To service a pending Send Byte Operation, the user writes a byte to the SMBDATA Register, which releases the SCL line high, and allows the Send Byte operation to continue. To service a pending Receive Byte Operation, the user reads a byte from the SMBDATA Register which releases the SCL line high.

### Address Comparator

The purpose of the Address Comparator is to give the SMBus Peripheral a method for determining if communication on the SMBus is intended for it. After every Start or Restart condition, the first 7 bits of data shifted in on SDA are compared to the values stored in the PRIMADDR and SECDADDR Address Registers. If there is a match with either of the two address registers, bit D7 of the SMBSTATUS Register will be set accordingly:

0 – SECDADDR Address Matched

1 – PRIMADDR Address Matched

If there is no match the SMBus control logic will reset to the Idle State.

### Bus Idle Detector

The Bus Idle Detector monitors the levels of the SCL and SDA signals to determine the current condition of the SMBus. Essentially, this circuit is trying to determine if the SMBus is free of traffic. According to the System Management Bus Specification 1.1 the SMBus is free when SCL and SDA are high for a period greater than 50µsec. If the Bus Idle Detector determines that the SCL and SDA signals have been high for a period greater than 50µsec, it will set the BUSIDLE bit (D5) in the SLOV Register.

### Master Clock Generator

The Master Clock Generator provides multiple clocks used by both the Transmitter and Receiver sections of the SMBus Block. It derives these clock signals from the core CPU's 3.5MHz instruction oscillator. During all Master operations and Slave send byte operations, the SMBus Interface requires the core CPU's 3.5MHz instruction oscillator. The SCL clock frequency can be programmed to operate at four different speeds by selecting bits D4 and D5 in the SMBCONFIG Register:

| SMBCONFIG | | Divider Ratio | Master SCL Frequency (Typical) |
|---|---|---|---|
| D5 | D4 | | |
| 0 | 0 | $f_{OSC}/72$ | 48.61 KHz |
| 0 | 1 | $f_{OSC}/168$ | 20.83 KHz |
| 1 | 0 | $f_{OSC}/296$ | 11.82 KHz |
| 1 | 1 | $f_{OSC}/488$ | 7.17 KHz |

The POR default Master SCL clock speed is 48.61 kHz.

### 35msec Timer

The 35msec Timer monitors the time that the SCL clock line is held LOW. The timer begins running whenever SCL is LOW and resets when SCL returns HIGH. Should SCL remain LOW longer than 35msec, the SMBus Interface logic will set the TOUT status bit in the SMBSTAT Register, and generate an interrupt to the processor core. If the SCL line is not released HIGH, this interrupt will continue to occur each 35msec. This safety feature insures that the SMBus may be reset in the event a transmission is not completed. Should this feature not be desired, the user can disable it by clearing the TOUT mask bit (D5) in the SMBMASK Register.

## Slave Mode Operation

Prior to operation, the User should enter values for both SMBus device addresses. When these addresses are set, the SMB Slave is ready to receive data independent of the MAX1780 processor core operation. The SMB status (STOP, START, RESTART) can be monitored in the SMBSTATUS register, as well as the ACK being detected. These signals will generate interrupts via INT2 if the appropriate SMBMASK bits are set. The SCL line is automatically held low after the Master sends the 9th clock pulse (Slave Acknowledge), and will not be released unless the SMBDATA Register is read from or written to by the MAX1780 processor core.

The SMBus Interface has one address for the SMBDATA register, though there are actually two independent shift registers at the address (receive and transmit shift registers). The last valid data byte sent on the bus will always be in the SMBDATA register, though the microcontroller will only get the Acknowledge Interrupt if a matching address has been detected after the START was received. Writing to the SMBDATA address will load the transmit shift register, but not interfere with the data in the receive shift register.

### Initialization

Update SMBSPEED[1:0] bits in the SMBCONFIG Register to select a SCL frequency. Default is 48.61 KHz.

Set the CMD[2:0] bits in the SMBCONFIG Register to 111 (Slave Receive).

Update the PRIMADDR and SECDADDR Address Registers to define the Slave device address. If both device addresses are not required, enter the same address in both registers.

## Master Mode Operation

Master command operations are selected via the CMD[2:0] bits in the SMBCONFIG Register. These bits must be set prior to any master mode operation. Generally, a Master SMBus transaction will consist of a START signal followed by a Slave address byte, then a number of data bytes, and complete the operation with a STOP condition. The SCL clock is held low at the end of all master mode operations except for STOP. When any master mode operation is finished, the MSTDON bit will be set in the SMBSTATUS Register and provide an interrupt if the corresponding SMBMASK bit is set. The Master may receive data from the slave by writing a 0xFF byte (this will not corrupt data on the SMBus) and then reading the SMBDATA register once the operation is completed.

The SMBus Interface can detect the SCL line extended low (clock stretching), and will pause the execution of the current Master operation until the SCL line is released high. At this time the Master will continue the last operation from where it was paused. If the SCL line is held low for longer than 35msec, the current Master operation will be terminated, the TOUT flag will be set in the SMBSTATUS Register, and an interrupt will be generated.

Master Mode operations require the MAX1780 Instruction Oscillator to be running in order to function properly. The user should insure that the OSLB bit in the OPTION Register is set to '1' so that the instruction oscillator will continue to run when the processor core is in SLEEP mode. This is only necessary for SMBus Master operations, where the MAX1780 must generate the SCL clock.

### Sending a START Signal

A START signal is defined as a high to low transition of SDA while SCL is high. A START will be initiated only if both SDA and SCL have been high for a period exceeding 50uS. The following steps explain how to generate the START signal:

1. Set the CMD[2:0] bits in the SMBCONFIG Register to 0b000, a **Send Start** operation.
2. Perform a "dummy write" 0xFF to the SMBDATA Register. This triggers the Master operation.
3. Wait for interrupt 2 or poll the SMBSTAT Register until the MSTDON flag is set.

### Sending the Slave Address and Data Direction Bit

The first data byte immediately after the START signal, or repeated START signal, contains the address of the Slave device. This is a seven bit long address followed by a data direction bit (R/W-bit). The R/W-bit tells the slave the desired direction of data transfer. Only a Slave device with a matched address will respond by sending back an acknowledge bit by pulling SDA low on the 9th clock cycle. The following steps explain how to send the first byte of data (slave address):

1. Set the CMD[2:0] bits in the SMBCONFIG Register to 0b100, a **Send Byte** operation. Also, set the **ACK/NACK** control bit D3 to '1' (NACK), so that the Slave Acknowledge can be detected.
2. Load the "W" Register with the Slave Address and Data Direction Bit to be moved to the SMBDATA Register.
3. Move the contents of the "W" Register to the SMBDATA Register. This triggers the Master operation.
4. Wait for interrupt 2 or poll the SMBSTAT Register until the MSTDON flag is set.
5. Check the SMBSTAT Register for ACKDET flag to be set, indicating a Slave Acknowledge.
6. Read back the byte from SMBDATA Register.
7. Check the byte sent for integrity.

### Sending a STOP Signal

The master can terminate an SMBus transaction by generating a STOP signal. A STOP signal is defined as a low to high transition of SDA while SCL is at logical high. The following steps explain how a STOP condition is generated by the Master transmitter:

1. Set the CMD[2:0] bits in the SMBCONFIG Register to 0b010, a **Send Stop** operation.
2. Perform a "dummy write" of 0xFF to the SMBDATA Register. This triggers the Master operation.
3. Wait for interrupt 2 or poll the SMBSTAT Register until the MSTDON flag is set.

### Sending a Repeated START Signal

A repeated START signal is used to generate a START signal without first generating a STOP signal to terminate the communication. This is used by the Master to indicate to the Slave device that the data direction will change (transmit/receive mode) without releasing the bus. A program example is shown below.

1. Set the CMD[2:0] bits in the SMBCONFIG Register to 0b011, a **Send Restart** operation.
2. Perform a "dummy write" of 0xFF to the SMBDATA Register. This triggers the Master operation.
3. Wait for interrupt 2 or poll the SMBSTAT Register until the MSTDON flag is set.

## SMBus Interface Register Descriptions

### SMBCONFIG Register (Read/Write): PORTB Address = 0x00

| Bit | Name | POR | Function/Description | | | | |
|---|---|---|---|---|---|---|---|
| D7 | | 0 | RFU | | | | |
| D6 | | 0 | RFU | | | | |
| D5-D4 | SMBSPEED[1:0] | 00 | D5 | D4 | Divider Ratio | SCL (Typ) | Units |
| | | | 0 | 0 | $f_{OSC}/72$ | 48.61 | KHz |
| | | | 0 | 1 | $f_{OSC}/168$ | 20.83 | KHz |
| | | | 1 | 0 | $f_{OSC}/296$ | 11.82 | KHz |
| | | | 1 | 1 | $f_{OSC}/488$ | 7.17 | KHz |
| D3 | ACKNACK | 0 | Master or Slave Mode:<br>0 = ACK the next byte sent.<br>1 = NACK the next byte sent. | | | | |
| D2-D0 | CMD[2:0] | 000 | Programs the Master Command Sequencer:<br>000 = Master Generate Start Condition<br>001 = Master Receive Byte<br>010 = Master Generate Stop Condition<br>011 = Master Generate Repeated Start Condition<br>100 = Master Send Byte<br>101 = Slave Send Byte<br>111 = Slave Receive | | | | |

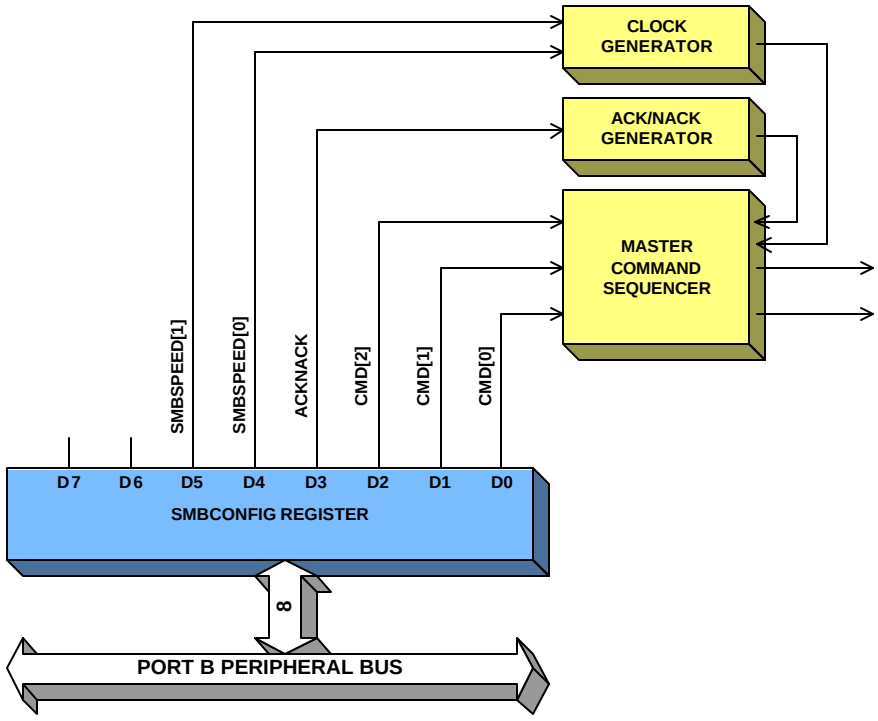



**Figure 23, SMBus Configuration Register**

### SMBSTATUS Register (Read/Write): PORTB Address = 0X01

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | ADDRSTAT | - | Only valid after receiving the first ACKNOWLEDGE of a valid address.<br>0 – Secondary Address Matched.<br>1 – Primary Address Matched. |
| D6 | SCLHOLD | 0 | Master or Slave Mode:<br>Set to 1 when SCL is held LOW after ACK.<br>Auto clear at next Read/Write of SMBDATA operation or at POR. |
| D5 | TOUT | 0 | Master or Slave Mode:<br>Set to 1 when SMBWDT timer exceeds the period (35mSec).<br>Auto clear at next Read/Write of SMBDATA operation or at POR. |
| D4 | MSTDON | 0 | Master Mode Only:<br>Set to 1 at the end of Any Master command operation.<br>Auto clear at next Read/Write of SMBDATA operation or at POR. |
| D3 | ACKDET | 0 | Master or Slave Mode:<br>Set to 1 when a valid acknowledge pulse is detected.<br>Auto clear at next Read/Write of SMBDATA operation or at POR. |
| D2 | STOPDET | 1 | Will be set to 1 each time a Stop Condition is detected on the SMBus.<br>Write 1 to Clear this bit |
| D1 | RESTARTDET | 0 | Will be set to 1 each time a Repeated Start Condition is detected on the SMBus.<br>Write 1 to Clear this bit. |
| D0 | STARTDET | 0 | Will be set to 1 each time a Start Condition is detected on the SMBus.<br>Write 1 to Clear this bit. |

Special Notes:

- Bits D6 through D0 are routed to INT2 and controlled by the SMBMASK register.
- Bits D7 and D6 are bus monitor bits. They are not cleared or set except by the state of the SCL and SDL lines.
- Bits D2, D1, and D0 are set when a STOP, RESTART, or START condition occurs. They are cleared by writing to the SMBSTATUS register. If the bits are not cleared “promptly”, it is possible to have more than one bit (D2, D1, D0) set at a time.
- The TOUT timer (bit D5) is running whenever the SCL pin is low in Slave Mode or a Master Mode transmit operation is in progress.

**SMBMASK Register (Read/Write):** **PORTB Address = 0x02**

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | | 0 | RFU |
| D6 | SCLHMSK | 0 | Master or Slave Mode:<br>0 = Mask SCLHOLD interrupts.<br>1 = Enable SCLHOLD interrupts. |
| D5 | TOUTMSK | 0 | Master or Slave Mode:<br>0 = Mask TOUT interrupts.<br>1 = Enable TOUT interrupts. |
| D4 | MSTDONMSK | 0 | Master Mode only:<br>0 = Mask MSTDON interrupts.<br>1 = Enable MSTDON interrupts. |
| D3 | ACKMSK | 0 | Master or Slave Mode:<br>0 = Mask ACK Detector interrupts.<br>1 = Enable ACK Detector interrupts. |
| D2 | SPMSK | 0 | Slave Mode only:<br>0 = Mask STOP interrupts.<br>1 = Enable STOP interrupts. |
| D1 | REMSK | 0 | Slave Mode only:<br>0 = Mask RESTART interrupts.<br>1 = Enable RESTART interrupts. |
| D0 | STMSK | 0 | Slave Mode only:<br>0 = Mask START interrupts.<br>1 = Enable START interrupts. |

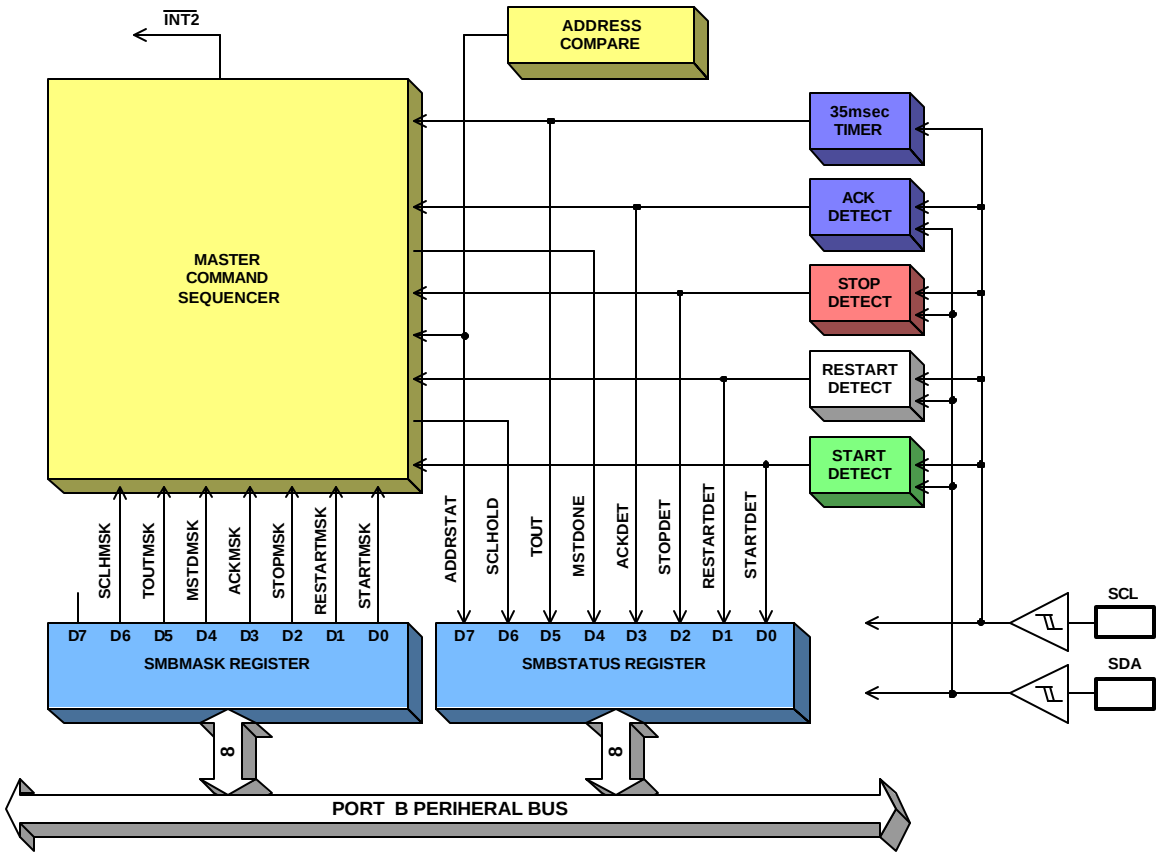



**Figure 24, SMBus Status And Mask Registers**

**SMBDATA Register (Read/Write):** **PORTB Address = 0x03**

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | SMB_D7 | - | SMBus Data Bit 7. |
| D6 | SMB_D6 | - | SMBus Data Bit 6. |
| D5 | SMB_D5 | - | SMBus Data Bit 5. |
| D4 | SMB_D4 | - | SMBus Data Bit 4. |
| D3 | SMB_D3 | - | SMBus Data Bit 3. |
| D2 | SMB_D2 | - | SMBus Data Bit 2. |
| D1 | SMB_D1 | - | SMBus Data Bit 1. |
| D0 | SMB_D0 | - | SMBus Data Bit 0. |

**PRIMADDR Latch (Write Only):** **PORTB Address = 0x04**

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | SMBADDR1_D7 | - | SMBus Address #1 Bit 7. |
| D6 | SMBADDR1_D6 | - | SMBus Address #1 Bit 6. |
| D5 | SMBADDR1_D5 | - | SMBus Address #1 Bit 5. |
| D4 | SMBADDR1_D4 | - | SMBus Address #1 Bit 4. |
| D3 | SMBADDR1_D3 | - | SMBus Address #1 Bit 3. |
| D2 | SMBADDR1_D2 | - | SMBus Address #1 Bit 2. |
| D1 | SMBADDR1_D1 | - | SMBus Address #1 Bit 1. |
| D0 | SMBADDR1_D0 | - | Not connected to Address Comparator |

Note: SMBADDR1 must be programmed before SMB slave mode operation will function.

**SECDADDR Latch (Write Only):** **PORTB Address = 0x05**

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | SMBADDR2_D7 | - | SMBus Address #2 Bit 7. |
| D6 | SMBADDR2_D6 | - | SMBus Address #2 Bit 6. |
| D5 | SMBADDR2_D5 | - | SMBus Address #2 Bit 5. |
| D4 | SMBADDR2_D4 | - | SMBus Address #2 Bit 4. |
| D3 | SMBADDR2_D3 | - | SMBus Address #2 Bit 3. |
| D2 | SMBADDR2_D2 | - | SMBus Address #2 Bit 2. |
| D1 | SMBADDR2_D1 | - | SMBus Address #2 Bit 1. |
| D0 | SMBADDR2_D0 | - | Not connected to Address Comparator |

Note: SMBADDR2 must be programmed before SMB slave mode operation will function.

**SLOV Register (Read/Write):** **PORTB Address = 0X06**

| Bit | Name | POR | Function/Description |
|---|---|---|---|
| D7 | SCLSTAT | - | State of the SCL pin |
| D6 | SDASTAT | - | State of the SDA pin |
| D5 | BUSIDLE | 0 | 1 = If both SCA and SCL have been high for 50µs. Read Only |
| D4 | SCLFORCE | 0 | 0 = SCL is controlled by the SMBus block circuitry.<br>1 = SCL is Asserted Low. |
| D3 | SDAFORCE | 0 | 0 = SDA is controlled by the SMBus block circuitry.<br>1 = SDA is Asserted Low. |
| D2 | PANIC | 0 | PANIC button (Reset the SMB state machine, Write Only) |
| D1 | BUSY | 0 | Set by Master Mode hardware (Read Only). |
| D0 | DETECT | 0 | State of the pin (Read Only) |

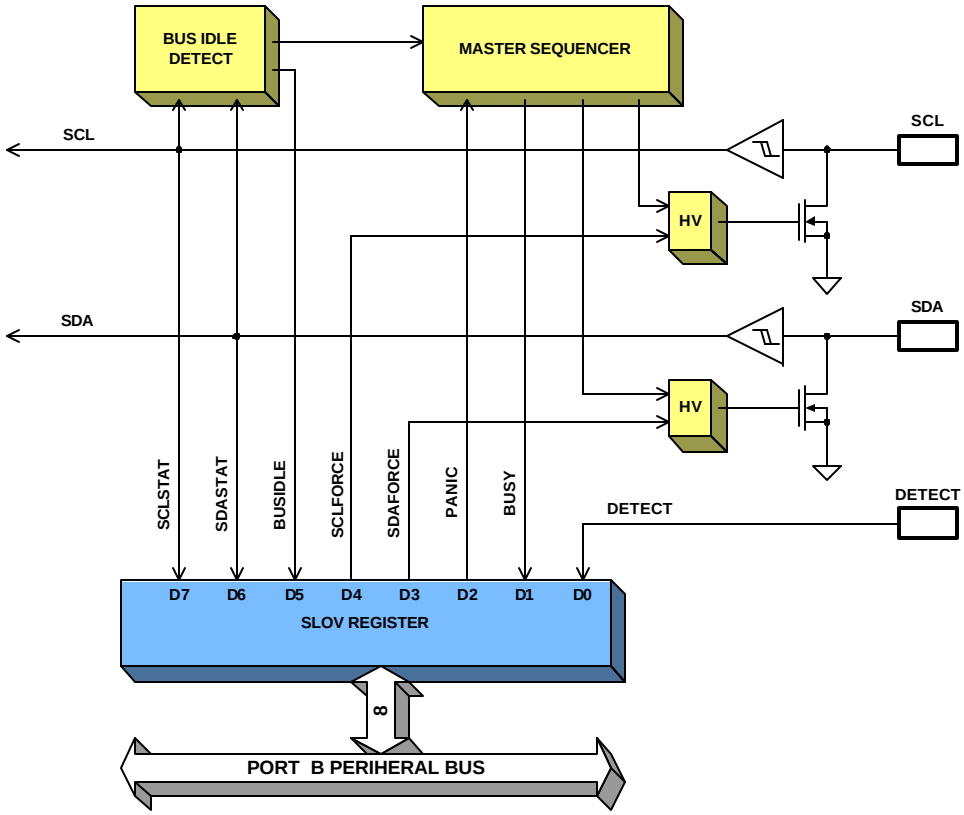



**Figure 25, SMBus SLOV Register**

# Design And Applications Information

## Interfacing With An External Serial EEPROM

The MAX1780 has been designed to directly interface with an external serial EEPROM. The high-speed SPI port can communicate with serial EEPROMs at SCLK frequencies of up to 4MHz. The MAX1780 typical operating circuit uses an 8K x 8, X25650 serial EEPROM capable of accepting a 4MHz SCLK signal. Figure 26 below shows how to interface a serial EEPROM to the MAX1780. Please refer to the latest X25650 data sheet, or to that of the selected serial EEPROM for detailed electrical specifications and proper operation.

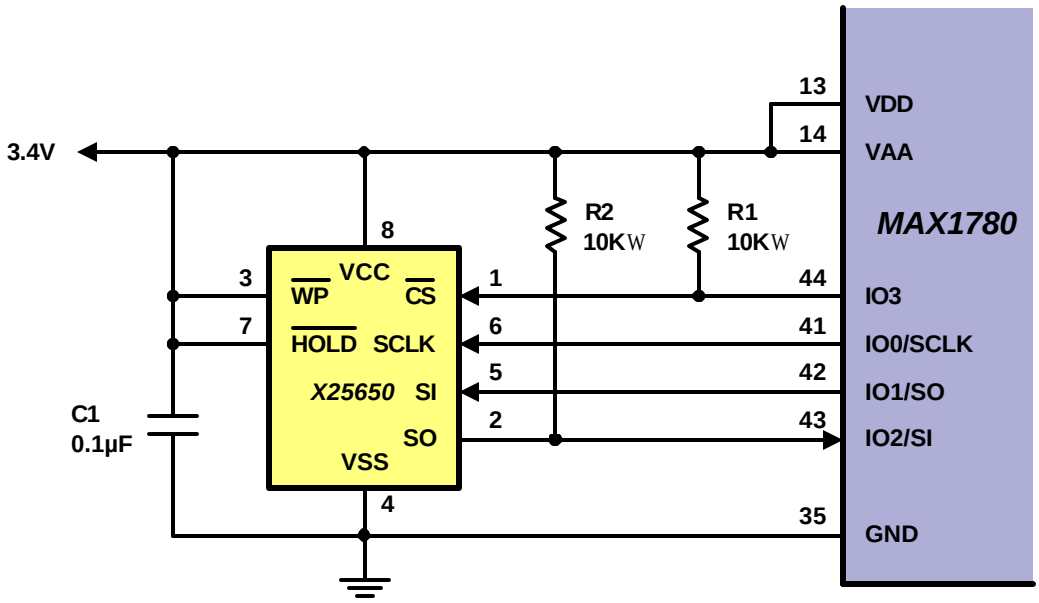



Figure 26, External EEPROM Interface Schematic

### Chip Select (CS\)

The Chip Select ($\overline{CS}$) signal may be any free GPIO pin on the MAX1780. The MAX1780 typical operating circuit however, makes use of IO3 for the $\overline{CS}$ signal. Connect the desired GPIO on the MAX1780 to the $\overline{CS}$ input pin of the EEPROM. Use a 10Kohm resistor from the $\overline{CS}$ pin to the EEPROM's VCC supply to pull the signal high. When $\overline{CS}$ is HIGH, the EEPROM is deselected and it's SO output pin is at high impedance.

### Serial Clock (SCLK)

The Serial Clock controls the serial bus timing for data input and output. The MAX1780 has a dedicated output pin (IO0/SCLK) for SCLK and should be connected to the SCLK input pin of the EEPROM.

### Serial Output (SO)

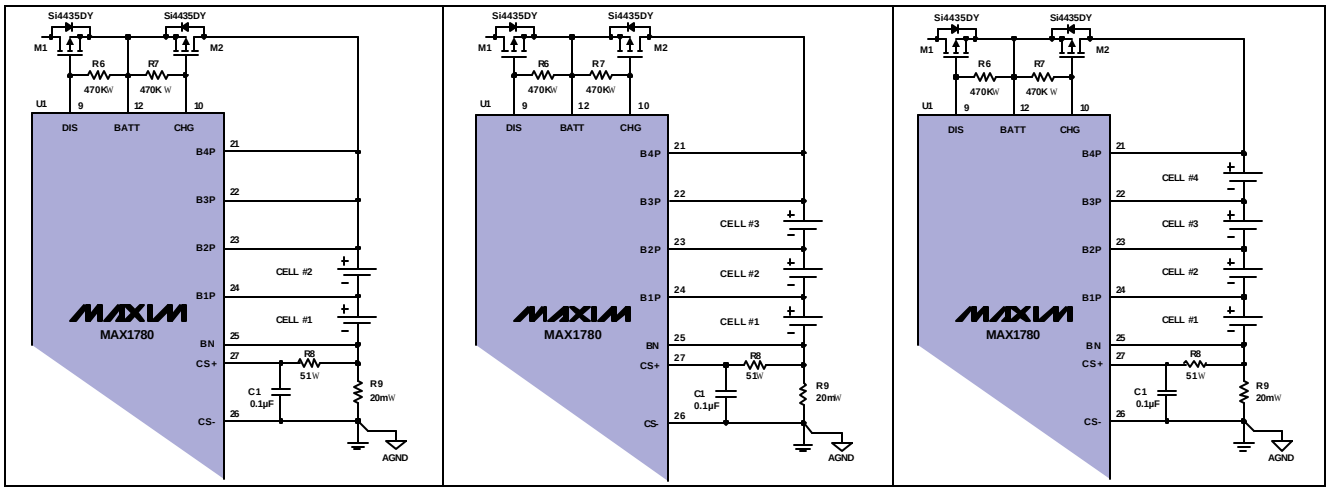
SO is a serial data output pin. The MAX1780 has a dedicated output pin (IO1/SO) and should be connected to the SI input pin of the serial EEPROM. During a write cycle, data is shifted out on this pin. Data from the MAX1780 SPI port is clocked out with the falling edge of SCLK.

### Serial Input (SI)

SI is the serial data input pin. The MAX1780 has a dedicated input pin (IO2/SI) and should be connected to the SO output pin of the serial EEPROM. Data into the MAX1780 SPI port is latched by the rising edge of SCLK. Use a 10K ohm resistor from the SI pin to the EEPROM's VCC supply to pull the signal high.

## Properly Connecting Lithium Ion Cells

The MAX1780 can directly connect to 2, 3, and 4 series Lithium Ion cell configurations. Figure 27 shows how to properly connect each of the series cell combinations. Note that in all cases, the positive terminal of the top series cell must have a connection to the B4P pin.




**2 Series Cells** **3 Series Cells** **4 Series Cells**

Figure 27, Proper Series Cell Connections

## Choosing The Current Sense Resistor ($R_{CS}$)

For greatest accuracy, choose $R_{CS}$ to ensure that the product of the maximum current to be measured $I_{MAX}$ and $R_{CS}$ does not exceed 137.33mV. Calculate the proper sense-resistor value as follows:

$$R_{CS} := \frac{V_{CS}}{I_{MAX}}$$

where $I_{MAX}$ is the maximum current to be accurately measured. Use only surface-mount metal-film resistors; wire-wound resistors are too inductive to provide acceptable results. Be sure to consider power dissipation when choosing the current-sense resistor to avoid resistor self-heating.

**Example**

Given:

$$I_{MAX} = 5 \cdot A$$

$$V_{CS} = 137.33 \cdot mV$$

Determining $R_{CS}$:

$$R_{CS} = \frac{V_{CS}}{I_{MAX}} \qquad R_{CS} = 0.027\Omega$$

Checking The $R_{CS}$ Power Dissipation:

$$P_{RCS} = \frac{V_{CS}^2}{R_{CS}} \qquad P_{RCS} = 0.687W$$

## Setting The Overcurrent Thresholds

**Overcharge Threshold**

Set the current at which the voltage on CS+ exceeds the voltage on OCI with a voltage divider placed between VAA and AGND (see Typical Operating Circuit). To set the overcharge threshold, choose R1 in the 1MΩ range and calculate R2 from:

$$R_2 = \frac{R_1}{\left(\frac{V_{AA}}{ICharge_{MAX} \cdot R_{CS}}\right) - 1}$$

where VAA = 3.4V, $ICharge_{MAX}$ is the maximum allowable charging current, and $R_{CS}$ is the current sense resistor value. As an example, suppose we want to open up the charge protection MOSFET whenever charge currents exceed 5.4A;

Given: $R_1 = 1 \cdot M\Omega$ $\qquad ICharge_{MAX} = 5.4 \cdot A$

$V_{AA} = 3.4 \cdot V$

$R_{CS} = 0.02 \cdot \Omega$

$$R_2 = \frac{R_1}{\left(\frac{V_{AA}}{ICharge_{MAX} \cdot R_{CS}}\right) - 1}$$

$R_2 = 32.807\ K\Omega$ Choosing the closest common value: $R_2 = 33 \cdot K\Omega$

Checking:

$$ICharge_{MAX} = R_2 \cdot \frac{V_{AA}}{R_{CS} \cdot (R_1 + R_2)}$$

$ICharge_{MAX} = 5.431\ A$

**Overdischarge Threshold**

Set the current at which the ODI voltage falls below AGND with a voltage divider placed between VAA and BN (see Typical Operating Circuit). To set the overdischarge threshold, choose R3 in the 1MΩ range then calculate R4 from:

$$R_4 = R_3 \cdot \left( \frac{IDischarge_{MAX} \cdot R_{CS}}{V_{AA}} \right)$$

where VAA = 3.4V, $IDischarge_{MAX}$ is the maximum allowable discharging current, and $R_{CS}$ is the current sense resistor value. As an example, suppose we want to open up the discharge protection MOSFET whenever discharge currents exceed 6.5A;

Given: $R_3 = 1 \cdot M\Omega$ $\quad IDischarge_{MAX} = 6.5 \cdot A$

$V_{AA} = 3.4 \cdot V$

$R_{CS} = 0.02 \cdot \Omega$

$$R_4 = R_3 \cdot \left( \frac{IDischarge_{MAX} \cdot R_{CS}}{V_{AA}} \right)$$

$R_4 = 38.235\,K\Omega$ Choosing the closest common value: $R_4 = 39 \cdot K\Omega$

Checking:

$$IDischarge_{MAX} = V_{AA} \cdot \frac{R_4}{(R_3 \cdot R_{CS})}$$

$IDischarge_{MAX} = 6.63\,A$

## Handling Battery Insertion Surge Currents

When inserting a battery pack into a notebook computer, an initial surge current will flow to charge up the notebook power supply's input capacitors. This large current spike can trigger the overdischarge current comparator, which will open up the Discharge MOSFET. This event will produce a nuisance Overcurrent interrupt. Figure 28 shows the MAX1780 overdischarge current detection circuitry, and the resistive elements that determine $I_{SURGE}$.

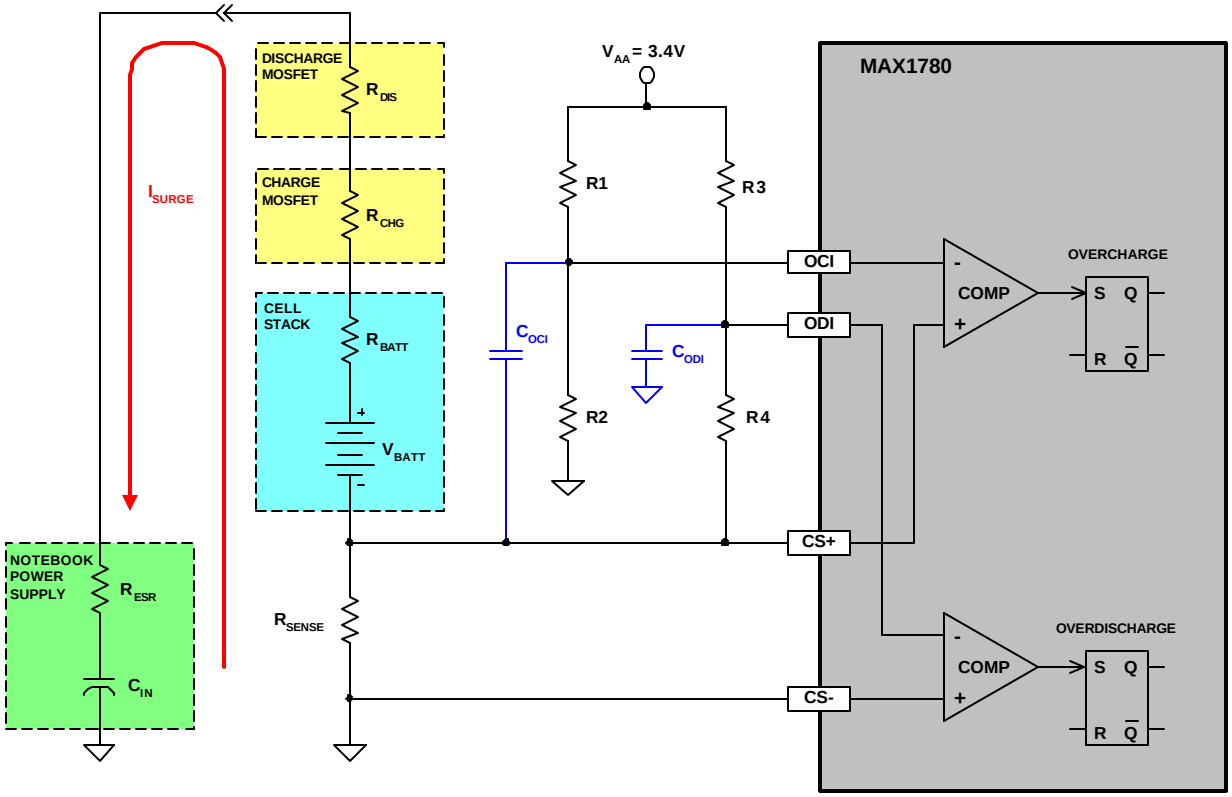



Figure 28, Overcurrent Comparator System Diagram

When the $I_{SURGE}$ current approaches its peak value, the voltage on the Overdischarge current comparator input ODI goes below ground. This will cause the Overdischarge current comparator to trip. To prevent this, connect a 100nF filter capacitor ($C_{ODI}$) from the ODI pin to AGND.

## Handling Charger Connection Surge Currents

When connecting a charger to a deeply discharged battery pack, a surge current will flow because of the extremely low impedance of the battery cells. This large current spike, if large enough, can trigger the overcharge current comparator, which will open up the Charge MOSFET. This event will produce a nuisance Overcurrent interrupt.

When the current approaches its peak value, the voltage on the Overcharge current comparator input OCI goes below ground. This will cause the Overcharge current comparator to trip. To prevent this, connect a 100nF filter capacitor ($C_{OCI}$) from the OCI pin to the CS+ pin.

## Improving Fuel Gauge Measurement Accuracy

### Use The Correct Ground Layout

The MAX1780 Fuel Gauge is very accurate, however care should be taken in laying out the PCB signal lines to the CS+ and CS- pins. Improper layout will result in degraded Fuel Gauge performance. Figure 29 below shows the recommended PCB layout. Vias to the ground plane should never be connected to either of the Kelvin traces that run between the sense resistor to the CS+ and CS- pins.

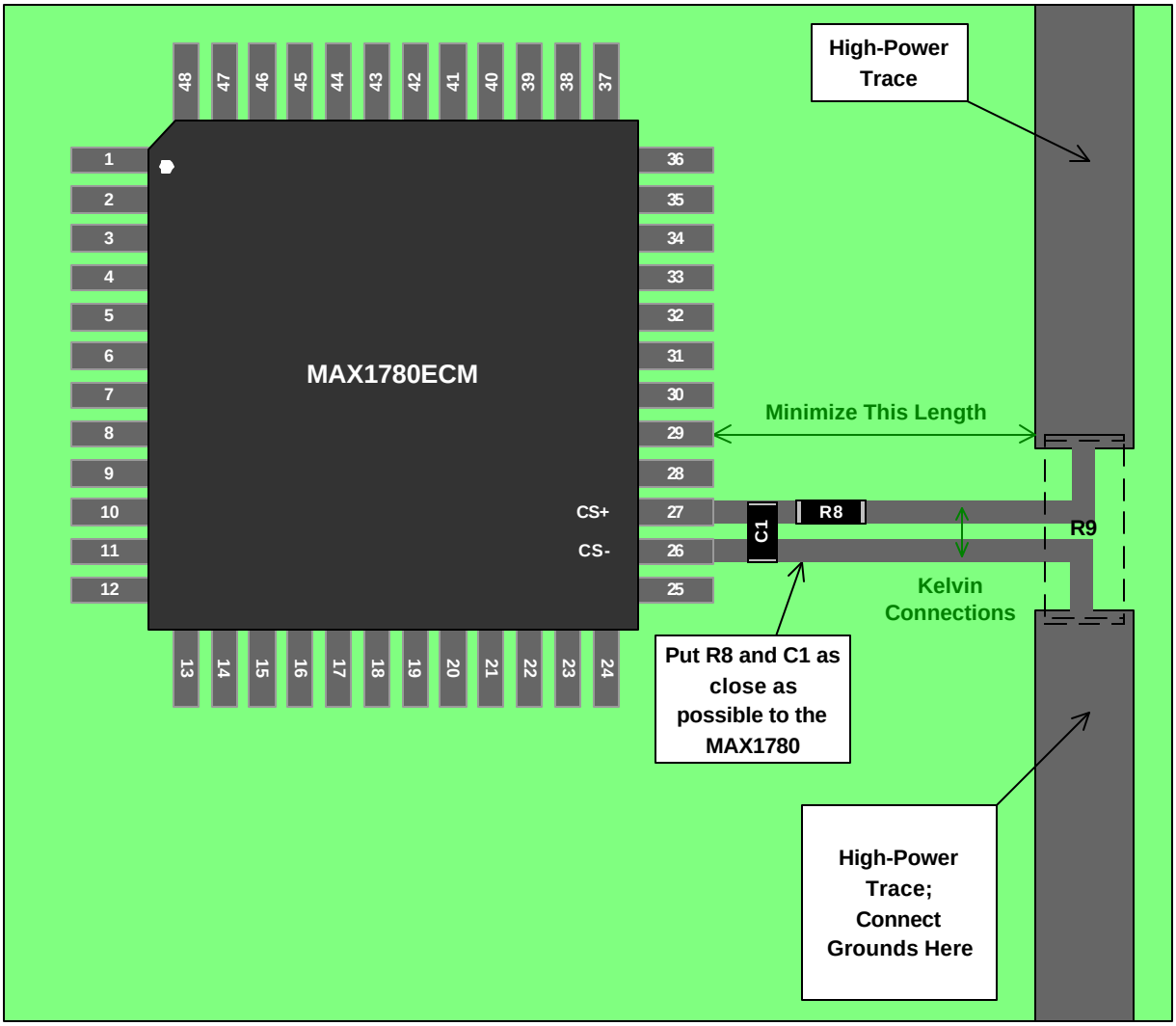



Figure 29, Layout Recommendation For Current Sense Inputs

### Filter The Current Sense Inputs

The high-power ground traces around the current sense resistor can be very noisy. This noise, if directly coupled to the Fuel Gauge will result in degraded performance. Therefore, place a 51Ω resistor between current sense resistor and CS+, and bypass CS+ to CS- with a 0.1µF ceramic capacitor (see Figure 29). This creates a low pass filter that greatly reduces the noise at the CS+ and CS- pins. To minimize leakage errors due to finite trace-to-trace resistance, place both filter components as close to the CS+ pin as possible.

## Connecting The $\overline{SHDN}$ Pin

Connect the $\overline{SHDN}$ pin to the positive battery pack terminal through a 3Megohm resistor (R5), as shown in Figure 33, MAX1780 Application Circuit. R5 limits the current consumed by the MAX1780 in shutdown. Connect a signal diode D1 (Cathode) from the $\overline{SHDN}$ pin to the IO4 pin (Anode). Bypass the $\overline{SHDN}$ pin to AGND with a 0.1µF capacitor. This reduces the charge injection caused by the Discharge MOSFET when it turns on.

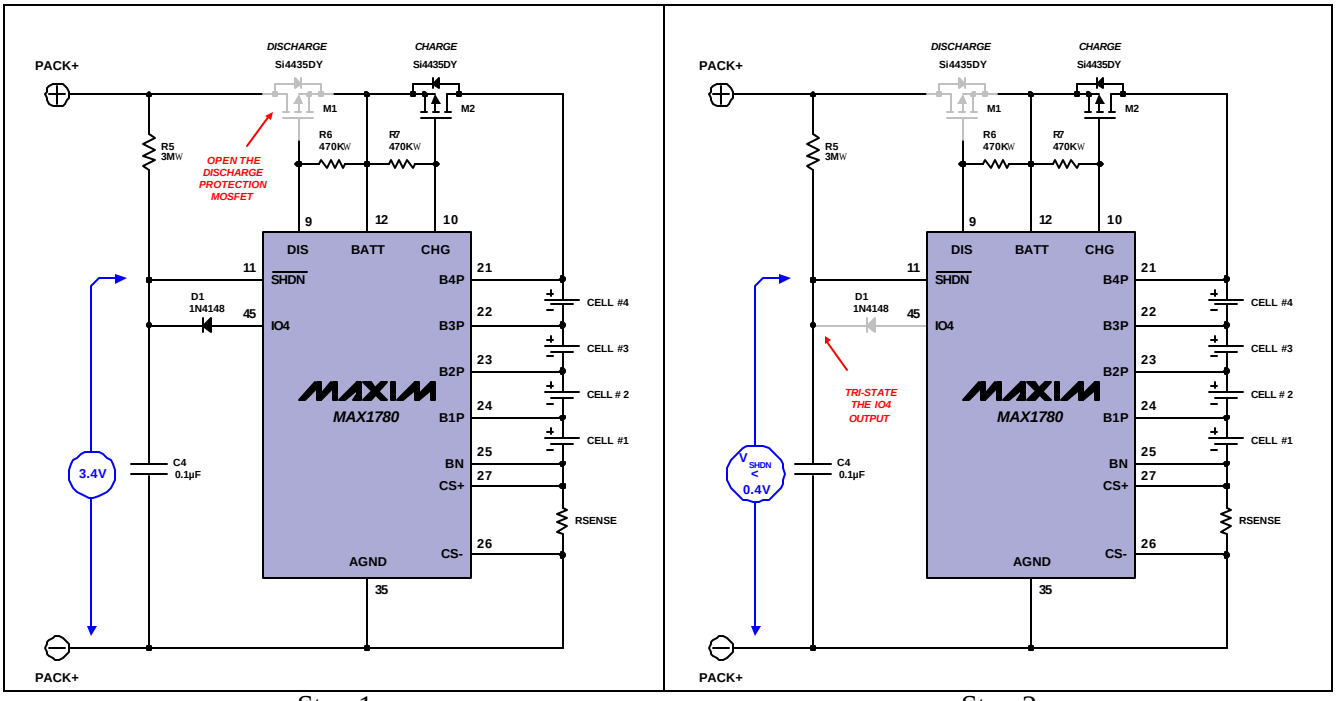
## Shutting Down The MAX1780 Under Software Control

Whenever the voltage on the $\overline{SHDN}$ pin is less than 0.4V, the MAX1780 is in Shutdown Mode. In shutdown, all CPU activity is stopped, the linear voltage regulator output (VAA) will be zero, and the MAX1780's current consumption will typically be less than 100µA. This low power state is excellent for minimizing cell discharge when storing MAX1780 based battery packs for extended periods. The following text provides MAX1780 software programmers with a simple procedure for implementing shutdown under software control. Please refer to Figure 30 below, and insure that the $\overline{SHDN}$ pin is connected as shown. Note that for shutdown to function under software control, any charge source connected to PACK+ and PACK- must be removed.

Users should develop their control software to continually measure individual cell voltages, and compare them with a defined minimum cell voltage threshold ($V_{CELL(MIN)}$). For Lithium Ion cells, this is typically 2.5V. Should any cell's voltage drop below $V_{CELL(MIN)}$ for a specified number of measurement cycles, the user software should perform the following tasks to shutdown the MAX1780:

1. Open the Discharge protection MOSFET.
2. Tri-state the IO4 GPIO pin.

Once the voltage on the $\overline{SHDN}$ pin has collapsed, the MAX1780 will shutdown. The MAX1780 will remain in this very low-power state until the startup procedure is followed.




Step 1 Step 2

Figure 30, MAX1780 Software Shutdown Procedure

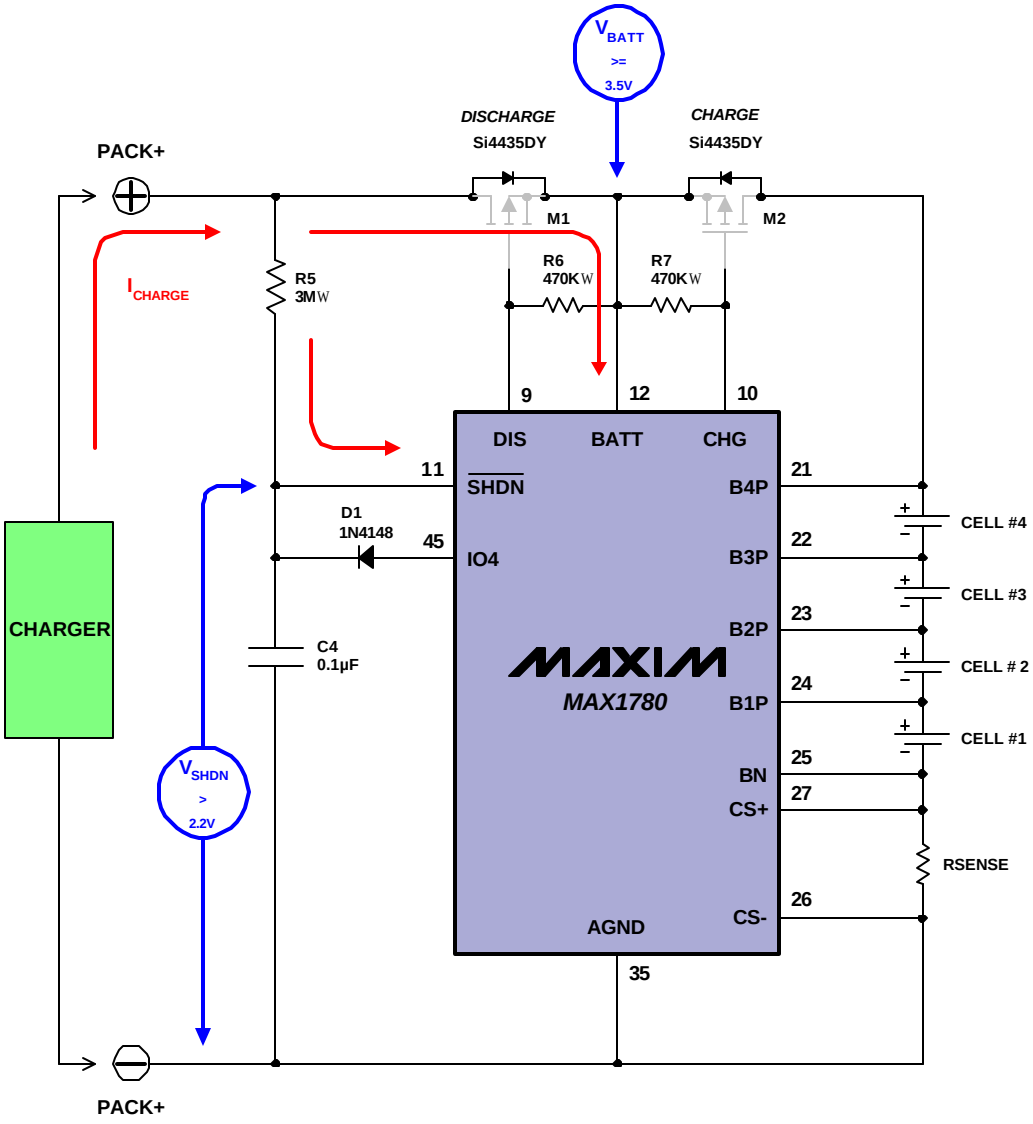



Figure 31, Shutdown Recovery Procedure

## Starting Up The MAX1780 After Being Shutdown

First, an external charge source is applied to the PACK+ and PACK- terminals. Current will enter through the Discharge MOSFET's body diode allowing the voltage on the BATT pin ($V_{BATT}$) to rise. Similarly, current will flow from the PACK+ terminal through the 3 Megohm dropping resistor allowing the voltage on the $\overline{SHDN}$ pin ($V_{SHDN}$) to rise. When $V_{SHDN}$ rises above 2.2V, the MAX1780 will initiate a Power-On/Reset, turning on the Charge and Discharge MOSFETs. At this point, the user's system software should begin measuring each cell's voltage, and comparing their values with the defined minimum cell voltage threshold $V_{CELL(MIN)}$. If all of the cell voltages are greater than or equal to this threshold value, the IO4 pin should be set to a logic '1'. This effectively "props up" the $\overline{SHDN}$ pin. The external charge source may now be removed. Now, even if the Discharge or Charge MOSFETs should be opened, the logic level produced by IO4 will cause the voltage on $\overline{SHDN}$ pin to be high enough to keep the MAX1780 from shutting down. The user's software should continue to check individual cell voltages, and as long as all of the cells are greater than the minimum cell voltage $V_{CELL(MIN)}$, allow normal operation to continue.

## Implementing An SBS-IF Safety Signal

In most SBS-IF compliant Smart Battery systems, a "Safety Signal" is used. This mechanism allows an independent communication path between the Smart Battery and the Smart Battery Charger to enhance the safety of the charging circuit. Essentially, the Safety Signal is connected to a circuit that provides three different resistances depending on the temperature of the battery pack. For a complete discussion of the Safety Signal implementation, please refer to the SBS-IF specifications for Smart Battery Chargers and Smart Batteries.

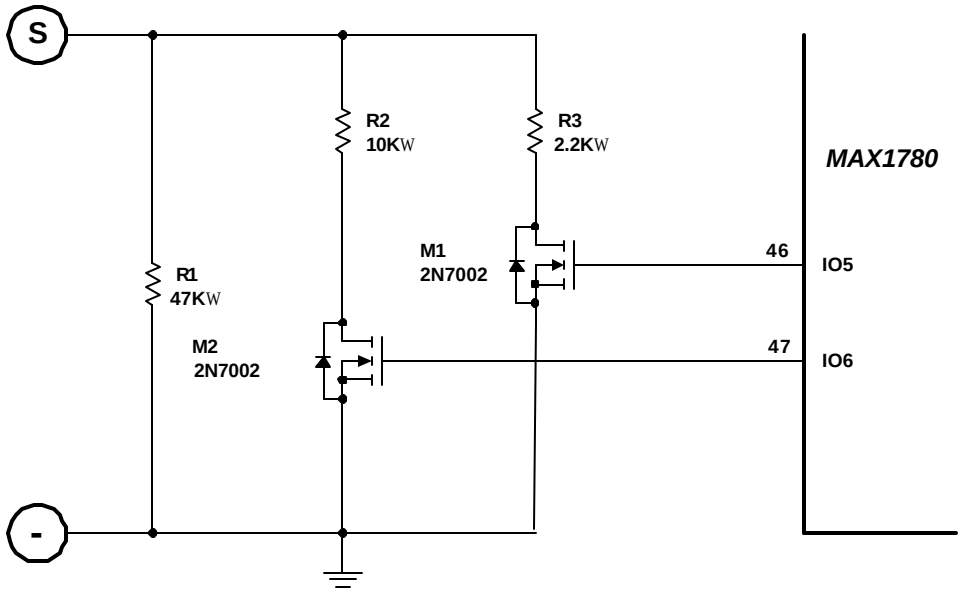



**Figure 32, MAX1780 Safety Signal Circuit**

Figure 32 provides a circuit for generating the Safety Signal using the IO5 and IO6 pins on the MAX1780. User developed Software running on the MAX1780 continually measures battery pack temperature to obtain the "Temp" value. It is compared against two user adjustable thresholds, "Cold" and "Hot" to determine the "Pack State". The IO5 and IO6 output pins are then driven in accordance to the following table:

| Temperature Comparison | Pack State | IO5 | IO6 |
|---|---|---|---|
| Temp ≥ Hot | Hot | 0 | 1 |
| Cold < Temp < Hot | Normal | 1 | 0 |
| Temp ≤ Cold | Cold | 0 | 0 |

## Circuit Layout And Grounding

Good PC board layout is essential to achieving the specified fuel gauge and data acquisition performance characteristics. Good layout includes the use of a ground plane, appropriate component placement, and correct routing of traces using appropriate trace widths. Design the printed circuit board so that the analog and digital sections are separated and confined to different areas on the board. Join the analog ground (AGND) and digital ground (GND) planes at a single-point star ground. Refer to the MAX1780 evaluation kit for an example of proper PCB layout.

# Pin Descriptions

| PIN | NAME | FUNCTION |
|---|---|---|
| 1-8 | HV0-HV7 | High Voltage I/O. Outputs are accessed via the HVO Register. The HV0 – HV5 pins are high voltage open drain outputs that pull down to GND. The HV6 and HV7 pins are high voltage open collector PNP outputs that pull up to BATT. |
| 9 | DIS | Over Discharge MOSFET drive output. Controls the P-channel MOSFET that prevents the battery from being deep discharged. Can also be used to disconnect battery stack from its external terminals when the pack is not installed in the computer. |
| 10 | CHG | Over Charge MOSFET drive output. Controls the P-channel MOSFET that prevents overcharge and also prevents fast charging when the battery is deeply discharged. |
| 11 | $\overline{\text{SHDN}}$ | Shutdown. High voltage shutdown input. |
| 12 | BATT | +4V to +28V Supply input. |
| 13 | VDD | Digital Supply Input. Connect to VAA for normal operation. |
| 14 | VAA | +3.4V Linear Regulator Supply Output. Connect to VDD for normal operation and bypass with a 0.47µF capacitor to AGND. |
| 15, 35 | AGND | Analog Ground. |
| 16 | GND | Digital Ground. |
| 17 | OSC1 | 32kHz crystal oscillator input. |
| 18 | OSC2 | 32kHz crystal oscillator input. |
| 19 | $\overline{\text{MCLR}}$ | Master Reset. Bypass with 10nF capacitor to GND. |
| 20 | AIN | User analog input to ADC. |
| 21 | B4P | Cell # 4 positive terminal. |
| 22 | B3P | Cell # 3 positive terminal. |
| 23 | B2P | Cell # 2 positive terminal. |
| 24 | B1P | Cell # 1 positive terminal. |
| 25 | BN | Cell # 1 negative terminal. |
| 26 | CS- | Kelvin current sense negative input for measuring battery current. |
| 27 | CS+ | Kelvin current sense positive input for measuring battery current. |
| 28 | ODI | Over discharge threshold input. |
| 29 | OCI | Over charge threshold input. |
| 30 | BTEST | Factory test input. Tie to GND. |
| 31-33 | BTM0-BTM2 | Factory test inputs. Tie to GND |
| 34 | DETECT | Edge-sensitive (rising or falling) interrupt input. It can be used to signal battery pack insertion to the CPU. |
| 36 | SDA | SMBus Data Line. Open drain DMOS output and logic sense input that is protected from shorts to the battery. |
| 37 | SCL | SMBus Clock Line. Open drain DMOS output and logic sense input that is protected from shorts to the battery. |
| 38 | N.C. | Not connected. |
| 39 | N.C. | Not connected. |
| 40 | EXTCLK | Factory test input. |

| | | |
|---|---|---|
| 41 | IO0/SCLK | GPIO Port C Bit D0 and when the SPI Interface is selected, functions as the SPI Master clock (SCLK) signal. |
| 42 | IO1/SO | GPIO Port C Bit D1 and when the SPI Interface is selected, functions as the SPI Master serial data output (SO) signal. |
| 43 | IO2/SI | GPIO Port C Bit D2 and when the SPI Interface is selected, functions as the SPI Master serial data input (SI) signal. |
| 44 | IO3 | GPIO Port C Bit D3. |
| 45 | IO4/TMRA | GPIO Port C Bit D4 and Timer A clock input. |
| 46 | IO5 | GPIO Port C Bit D5. |
| 47 | IO6 | GPIO Port C Bit D6. |
| 48 | IO7/INT1 | GPIO Port C Bit D7 and Interrupt 1 input |

# Detailed Operating Circuit

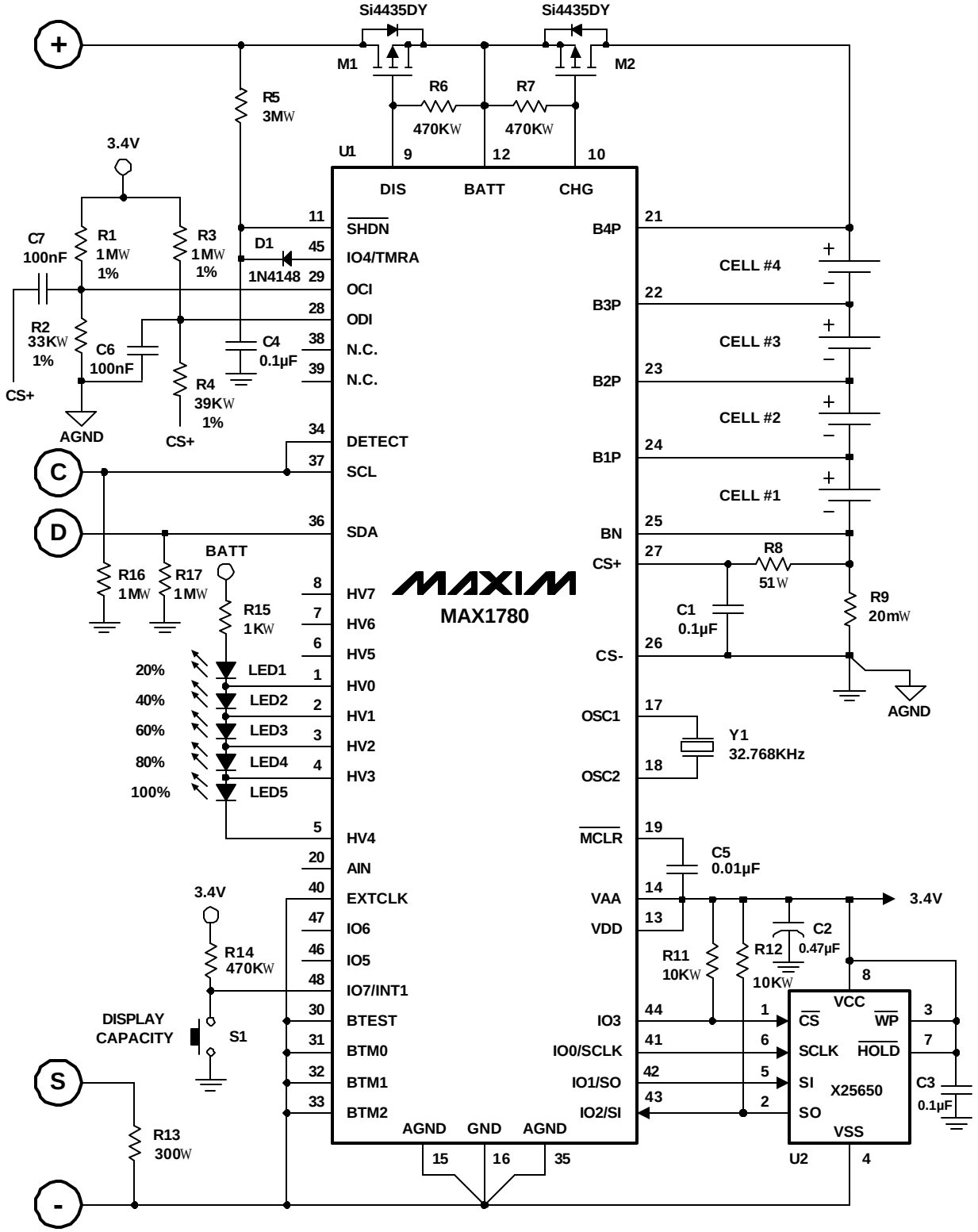



**Figure 33, MAX1780 Application Circuit**

# Absolute Maximum Ratings

VDD = VAA, AGND = GND, unless otherwise noted.

| PIN | MIN | MAX | UNITS |
|---|---|---|---|
| HV[5:0], $\overline{SHDN}$ , BATT, B1P, B2P, B3P, B4P, SCL, SDA, and DETECT to GND | -0.3 | 30 | V |
| VAA, VDD, and $\overline{MCLR}$ to GND | -0.3 | 6 | V |
| OSC1, OSC2, OCI, ODI, BTEST, BTM[2:0], EXTCLK, IO[7:0] to GND | -0.3 | VDD+0.3 | V |
| CS+, AIN, and BN to GND | -2 | 6 | V |
| CS- to GND | -1 | 1 | V |
| AGND to GND, and VAA to VDD | -0.3 | 0.3 | V |
| DIS, CHG, HV[7:6] to GND | -0.3 | BATT+0.3 | V |
| B3P, B2P, B1P to GND | -0.3 | B4P +0.3 | V |

*Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications in not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

# Electrical Characteristics

BATT=12V, AGND=GND=CS-=0V, VDD=VAA, $\overline{SHDN}$ =3.4V, $C_{VAA}$=0.47 µF, INSTOSC=OFF, 32.768kHz on OSC2 **$T_A$ = 0°C to +85°C**, unless otherwise noted.

| PARAMETER | CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| BATT LINEAR REGULATOR (VAA) | | | | | |
| Input Voltage Range | Includes dropout operation | 3.5 | | 28 | V |
| Output Voltage | 4V < BATT < 28V, 0 < $I_{LOAD}$ < 10mA | 3.2 | 3.4 | 3.6 | V |
| VAA Leakage Current | BATT = 0V, VAA = 3.4V | | 0.1 | 1 | mA |
| Short Circuit Current | VAA = 0V, 2.5V < BATT < 28V | 15 | 40 | 120 | mA |
| Linear Regulator, Power Fail, and 32kHz Clock Supply Current in Drop-out | BATT = 3.4V | | 24 | 150 | µA |
| Power Fail Warning Trip Level ($\overline{PFW}$) | Falling VAA | 2.91 | 3.0 | 3.09 | V |
| | Hysteresis on rising VAA | | 46 | | mV |
| Power On Reset Trip Level ($\overline{MCLR}$) | Rising VAA | 2.62 | 2.7 | 2.78 | V |
| | Hysteresis on falling VAA | | 41 | | mV |
| SUPPLY CURRENT (BATT) | | | | | |
| SHUTDOWN: Everything Off | $\overline{SHDN}$ < 0.4V | | 0.001 | 1 | µA |
| SLEEP: Linear Regulator, Power Fail, and 32kHz Clock. | INSTOSC = OFF | | 22 | 40 | µA |
| | INSTOSC = ON | | 330 | 550 | µA |
| FG + OD/OC: Linear Regulator, Power Fail, 32kHz Clock, Fuel Gauge, Precision Reference, OD/OC Comparators & MOSFET Drivers active. | CPU RUNNING | | 0.72 | 1.2 | mA |
| | CPU SLEEP (INSTOSC=OFF) | | 93 | 160 | µA |

# Electrical Characteristics (continued)

BATT=12V, AGND=GND=CS-=0V, VDD=VAA, $\overline{SHDN}$ =3.4V, $C_{VAA}$=0.47 µF, INSTOSC=OFF, 32.768kHz on OSC2 **$T_A$ = 0°C to +85°C**, unless otherwise noted.

| PARAMETER | CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| SUPPLY CURRENT (BATT) | | | | | |
| AFE/ADC + FG + OD/OC: Linear Regulator, Power Fail, 32kHz Clock, Fuel Gauge, Precision Reference, OD/OC Comparators & MOSFET Drivers active. | CPU RUNNING | | 1.4 | 2.4 | mA |
| | CPU SLEEP (INSTOSC=OFF) | | 0.82 | 1.3 | mA |
| Cancellation Circuit Supply Current from B4P (Only present during ADC conversions). | One cancellation circuit on. 15V applied to cancellation pin. | | 65 | 120 | µA |
| | Two cancellation circuits on. 15V applied to cancellation pins. | | 130 | 240 | µA |
| FUEL GAUGE | | | | | |
| Input Voltage Offset | | | 1 | 30 | µV |
| Input Resistance | CS+ to AGND | | 113 | | kΩ |
| Charge Coulomb-Counter Accumulation Rate | $V_{CS}$ = 137.33mV | 49,500 | 50,000 | 50,500 | Counts/s |
| Discharge Coulomb-Counter Accumulation Rate | $V_{CS}$ = -137.33mV | 49,500 | 50,000 | 50,500 | Counts/s |
| DATA ACQUISITION UNIT *(Offset Cancellation Enabled)* | | | | | |
| Conversion Time | 11-Bit | | 4.5 | | ms |
| | 13-Bit | | 17 | | |
| | 15-Bit | | 66 | | |
| | 16-Bit | | 132.95 | | |
| LSB Bit weight | High Voltage = 20.48V | | 312.50 | | µV |
| | Low Voltage = 5.12V | | 78.126 | | |
| Gain Accuracy at Full Scale | VAA: Tested @VDD=VAA=BATT=3.4V | -1 | | 1 | % |
| | BN, AIN: Tested @ BN/AIN to AGND=5.12V | -1 | | 1 | % |
| | B1P through B4P referenced to BN | 0.8 | | 0.8 | % |
| Integral Non linearity (INL) | | | 4 | | LSBs |

# Electrical Characteristics (continued)

BATT=12V, AGND=GND=CS-=0V, VDD=VAA, $\overline{SHDN}$ =3.4V, $C_{VAA}$=0.47 µF, INSTOSC=OFF, 32.768kHz on OSC2 **$T_A$ = 0°C to +85°C**, unless otherwise noted.

| PARAMETER | CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| DATA ACQUISITION UNIT *(Offset Cancellation Enabled)* | | | | | |
| Input Offset Voltage | | | 0.1 | 1 | mV |
| Input Voltage Range | B1P through B4P referenced to AGND | -0.2 | | 20.48 | V |
| | AIN, BN referenced to AGND | -2 | | 5.12 | V |
| Input Resistance to AGND with ADC (ON): | Low Voltage Range: AIN, BN | 0.5 | 1 | 1.8 | Meg Ohm |
| | High Voltage Range: BN, B1P, B2P, B3P, B4P No input bias current cancellation. | 2 | 4 | 7.2 | Meg Ohm |
| Input Bias Current with ADC & Input Bias Cancellation (ON): (B3P, B2P, B1P) | B4P = 20V, and B3P=B2P=B1P=15V referenced to AGND | | 30 | | nA |
| Input Bias Current with ADC (OFF): | B4P=B3P=B2P=B1P=20.48V referenced to AGND | | 0 | | nA |
| Input Bias Current with ADC (OFF): | BN=AIN=5.12V referenced to AGND | | 0 | | nA |
| Common Mode Error: B4P-B3P, B3P-B2P, and B2P-B1P | B4P=B3P=B2P=B1P=20.48V referenced to AGND Note: This error is proportional to the common mode voltage from AGND. | | 6 | 36 | mV |
| ADC Cell Differential Voltage Measurement Accuracy: | | | | | |
| B3P to B4P | B3P = 12.75V, B4P = 17V | -50 | 0 | +50 | mV |
| B2P to B3P | B2P = 8.5V, B3P = 12.75V | -50 | 0 | +50 | mV |
| B1P to B2P | B1P = 4.25V, B2P = 8.5V | -50 | 0 | +50 | mV |
| BN to B1P | BN = 0V, B1P = 4.25V | -50 | 0 | +50 | mV |
| ON-CHIP TEMPERATURE SENSOR *(Offset Cancellation Enabled)* | | | | | |
| LSB bit weight | | | 0.0078125 | | °K |
| Temperature Measurement Error | | | +/-4 | | °K |

# Electrical Characteristics (continued)

BATT=12V, AGND=GND=CS-=0V, VDD=VAA, $\overline{SHDN}$ =3.4V, $C_{VAA}$=0.47 µF, INSTOSC=OFF, 32.768kHz on OSC2 **$T_A$ = 0°C to +85°C**, unless otherwise noted.

| PARAMETER | CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| OVERCHARGE AND OVERDISCHARGE COMPARATORS | | | | | |
| ODI, OCI Input Offset Voltage | | -7 | | 7 | mV |
| ODI, OCI Input Bias Current | FET input comparators | -1 | | 1 | µA |
| ODI, OCI Comparator Propagation Delay | 20mV input overdrive | 3 | | 30 | µs |
| DIS, CHG Sink Current | DIS, CHG = 2V | 20 | | 80 | µA |
| DIS, CHG Sink Current | DIS, CHG = 20V, BATT = 20V | 30 | | 100 | µA |
| DIS, CHG Source Current | 0V < DIS, CHG < BATT-2V | 4 | 12 | | mA |
| Leakage Current | DIS = CHG = 28V | | | 1 | µA |
| INSTRUCTION OSCILLATOR | | | | | |
| Frequency | -40°C to +85°C, and 2.8V < VDD < 3.6V | 3 | 3.5 | 4 | MHz |
| Temperature Variation | -40°C to +85°C | | -2 | | % |
| Supply Sensitivity | 2.8V < VDD < 3.6V | | 2.4 | 8 | % |
| 32KHZ TIMER OSCILLATOR | | | | | |
| OSC1 Input Current | OSC1=0V or 3.4V, OSC2 is floating | | 125 | 500 | nA |
| OSC2 Sink Current | OSC1= OSC2 = 3.4V | 4 | 10 | 20 | µA |
| OSC2 Source Current | OSC1= OSC2=0V | 3 | 7 | 20 | µA |
| Transconductance | | 1.3 | 5 | 12 | µA/V |
| LOGIC INPUTS/OUTPUTS | | | | | |
| IO[7:0] Input Voltage Low | | | | 0.8 | V |
| IO[7:0] Input Voltage High | | 2.4 | | | V |
| IO[7:0] High Impedance Leakage Current | I/O pins programmed to Hi-Z. | | 0 | 1 | µA |
| IO[7:0] Output Voltage Low | $I_{SINK}$ = 2mA | | | 0.4 | V |
| IO[7:0] Output Voltage High | $I_{SOURCE}$ = 2mA | VDD-0.6 | | | V |
| $\overline{MCLR}$ RESET PIN | | | | | |
| $\overline{MCLR}$ Output Voltage Low | $I_{SINK}$ = 1mA, and BATT = VAA = VDD = 2.5V | | | 0.4 | V |
| $\overline{MCLR}$ Output Voltage High | $I_{SOURCE}$ = 0 BATT = VAA = VDD = 3.4V | VDD - 0.2 | | | V |
| $\overline{MCLR}$ Pull-up Resistance | Internally connected from $\overline{MCLR}$ to VDD. | 10 | 20 | 40 | kΩ |
| $\overline{MCLR}$ Input High Voltage | | 2.4 | | | V |
| $\overline{MCLR}$ Input Low Voltage | | | | 1 | V |
| $\overline{MCLR}$ Input Hysterisis | | | 250 | | mV |

# Electrical Characteristics (continued)

BATT=12V, AGND=GND=CS-=0V, VDD=VAA, $\overline{SHDN}$ =3.4V, $C_{VAA}$=0.47 µF, INSTOSC=OFF, 32.768kHz on OSC2 **$T_A$ = 0°C to +85°C**, unless otherwise noted.

| PARAMETER | CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| HIGH-VOLTAGE LOGIC INPUT (DETECT) | | | | | |
| DETECT Input High Voltage | | 2.1 | | | V |
| DETECT Input Low Voltage | | | | 0.8 | V |
| DETECT Output High Leakage Current | $V_{DETECT}$ = 28V | | | 1 | µA |
| SHUTDOWN PIN LOGIC LEVELS ($\overline{SHDN}$ ) | | | | | |
| Input High Voltage | | 2.2 | | | V |
| Input Low Voltage | | | | 0.4 | V |
| Input Bias Current | $\overline{SHDN}$ = 3.6V | | 0.8 | 2.0 | µA |
| | $\overline{SHDN}$ = 28V | | 18 | 50 | µA |
| HIGH-VOLTAGE OPEN DRAIN OUTPUTS THAT PULL TO GND (HV0-HV5) | | | | | |
| Output Voltage Low | $I_{SINK}$ = 7 mA | | | 1.4 | V |
| Leakage Current | Output High, $V_{APPLIED}$ = 28V | | | 1 | µA |
| HIGH-VOLTAGE OPEN COLLECTOR OUTPUTS THAT PULL TO BATT (HV6 and HV7) | | | | | |
| Output Voltage High | $I_{SOURCE}$ = 100µA | BATT-0.5 | | | V |
| Leakage Current | HV6 = HV7 = 0V | | | 1 | µA |

# SPI Interface Electrical Characteristics

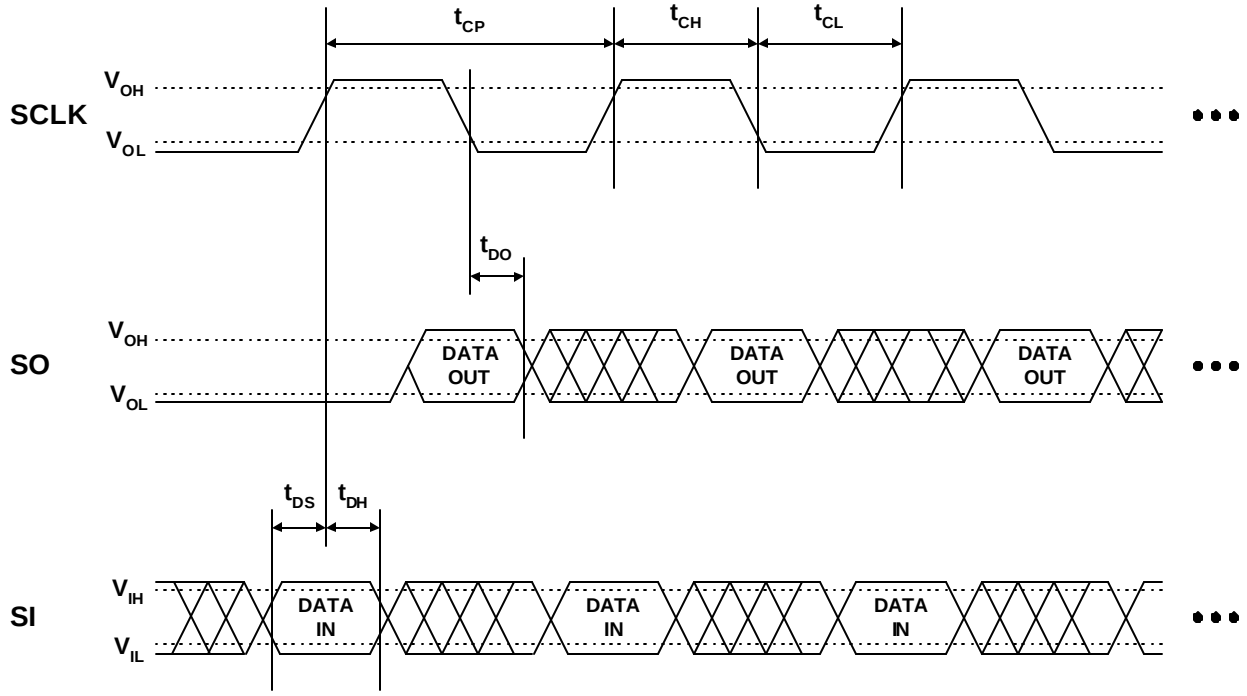



Figure 34, SPI Interface Timing

BATT=12V, AGND=GND=CS=0V, VDD=VAA, $\overline{SHDN}$ =3.4V, $C_{VAA}$=0.47 µF, INSTOSC=OFF, 32.768kHz on OSC2 **$T_A$ = 0°C to +85°C**, unless otherwise noted.

| SYMBOL | PARAMETER/CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| SPI INTERFACE AC TIMING *($C_{LOAD}$ = 20pF)* | | | | | |
| $t_{CP}$ | SCLK Period | 250 | | | ns |
| $t_{CH}$ | SCLK Pulse Width High | 100 | | | ns |
| $t_{CL}$ | SCLK Pulse Width Low | 100 | | | ns |
| $t_{DO}$ | SCLK Fall to SO Valid | -30 | | 30 | ns |
| $t_{DS}$ | SI to SCLK Data Setup Time | 70 | | | ns |
| $t_{DH}$ | SI to SCLK Data Hold Time | 0 | | | ns |
| SPI INTERFACE DC CHARACTERISTICS | | | | | |
| $V_{OH}$ | SPI Output High; $I_{SOURCE}$ = 2mA | VDD-0.6 | | | V |
| $V_{OL}$ | SPI Output Low; $I_{SOURCE}$ = 2mA | | | 0.4 | V |
| $V_{IH}$ | SPI Input High | 2.4 | | | V |
| $V_{IL}$ | SPI Input Low | | | 0.8 | V |

# SMBus Interface Electrical Characteristics

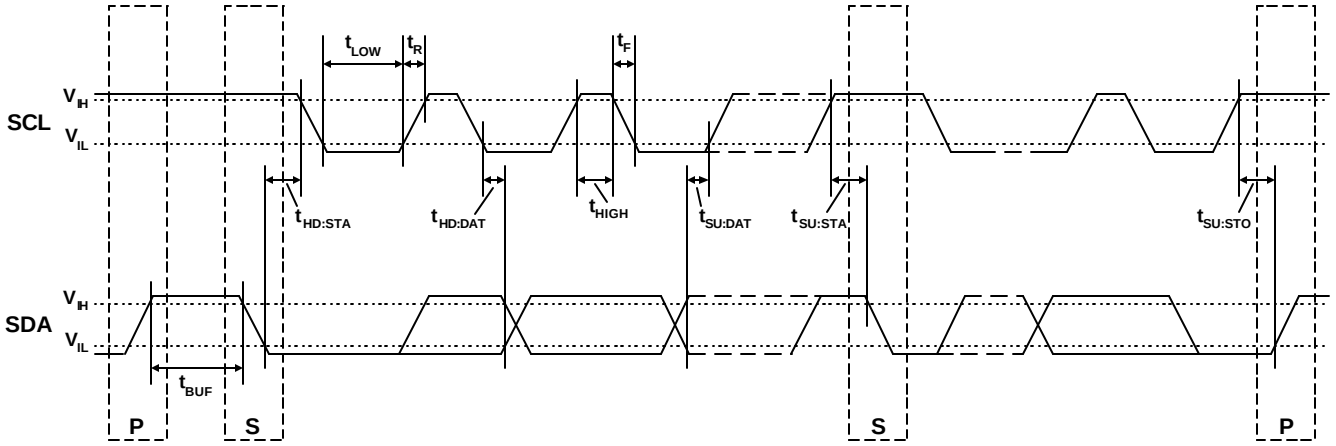



Figure 35, SMBus Timing Diagram

BATT=12V, AGND=GND=CS-=0V, VDD=VAA, $\overline{SHDN}$ =3.4V, $C_{VAA}$=0.47 µF, INSTOSC=OFF, 32.768kHz on OSC2 **$T_A$ = 0°C to +85°C**, unless otherwise noted.

| SYMBOL | PARAMETER/CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| Slave Operation | | | | | |
| $t_{BUF}$ | Bus free time between a Stop and Start condition. | 4.7 | | | µs |
| $t_{HD:STA}$ | Hold time after a (Repeated) Start condition. | 4.0 | | | µs |
| $t_{SU:STA}$ | Repeated Start setup time. | 4.7 | | | µs |
| $t_{SU:STO}$ | Stop condition setup time. | 4.0 | | | µs |
| $t_{HD:DAT}$ Data hold time. | Transmit | 300 | | | ns |
| | Receive | 0 | | | ns |
| $t_{SU:DAT}$ Data setup time. | Receive | 250 | | | ns |
| $t_{TIMEOUT}$ | Detect clock (SCL) low timeout. | | | 35 | ms |
| $t_{LOW}$ | Clock (SCL) low period. | 4.7 | | | µs |
| $t_{HIGH}$ | Clock (SCL) high period. | 4.0 | | | µs |

BATT=12V, AGND=GND=CS-=0V, VDD=VAA, $\overline{SHDN}$ =3.4V, $C_{VAA}$=0.47 µF, INSTOSC=OFF, 32.768kHz on OSC2 **$T_A$ = 0°C to +85°C**, unless otherwise noted.

| SYMBOL | PARAMETER/CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| All Master Operations | | | | | |
| $t_{TIMEOUT}$ | Detect clock (SCL) low timeout. | | | 35 | ms |
| Master Operation (SMBSPEED bits = 00) | | | | | |
| $f_{SCL}$ | SCL clock frequency. | 42.2 | 48.6 | 56.5 | KHz |
| $t_{HD:STA}$ | Hold time after a (Repeated) Start condition. | 8.2 | 9.7 | 11.6 | µs |
| $t_{SU:STA}$ | Repeated Start setup time. | 19.7 | 22.9 | 27 | µs |
| $t_{SU:STO}$ | Stop condition setup time. | 8.2 | 9.7 | 11.6 | µs |
| $t_{HIGH}$ | SCL High Period | 8.7 | 10.3 | 12.3 | µs |
| $t_{LOW}$ | SCL Low Period | 8.7 | 10.3 | 12.3 | µs |
| $t_{HD:DAT}$ Data hold time. | Master Transmit | 4.2 | 5.1 | 6.3 | µs |
| | Master Receive | 0 | | | ns |
| | Master Acknowledge | 300 | | | ns |
| $t_{SU:DAT}$ Data setup time. | Master Transmit | 4.2 | 5.1 | 6.3 | µs |
| | Master Receive | 250 | | | ns |
| | Master Acknowledge | 8.7 | 10.3 | 12.3 | µs |
| Master Operation (SMBSPEED bits = 01) | | | | | |
| $f_{SCL}$ | SCL clock frequency. | 17.96 | 20.83 | 23.81 | KHz |
| $t_{HD:STA}$ | Hold time after a (Repeated) Start condition. | 20.2 | 23.4 | 27.6 | µs |
| $t_{SU:STA}$ | Repeated Start setup time. | 49.7 | 57.1 | 67 | µs |
| $t_{SU:STO}$ | Stop condition setup time. | 20.2 | 23.4 | 27.6 | µs |
| $t_{HIGH}$ | SCL High Period | 20.7 | 24 | 28.3 | µs |
| $t_{LOW}$ | SCL Low Period | 20.7 | 24 | 28.3 | µs |
| $t_{HD:DAT}$ Data hold time. | Master Transmit | 10.2 | 12 | 14.3 | µs |
| | Master Receive | 0 | | | ns |
| | Master Acknowledge | 300 | | | ns |
| $t_{SU:DAT}$ Data setup time. | Master Transmit | 10.2 | 12 | 14.3 | µs |
| | Master Receive | 250 | | | ns |
| | Master Acknowledge | 20.7 | 24 | 28.3 | µs |
| Master Operation (SMBSPEED bits = 10) | | | | | |
| $f_{SCL}$ | SCL clock frequency. | 10.17 | 11.82 | 13.56 | KHz |
| $t_{HD:STA}$ | Hold time after a (Repeated) Start condition. | 36.2 | 41.7 | 49 | µs |
| $t_{SU:STA}$ | Repeated Start setup time. | 89.7 | 102.9 | 120.3 | µs |
| $t_{SU:STO}$ | Stop condition setup time. | 36.2 | 41.7 | 49 | µs |
| $t_{HIGH}$ | SCL High Period | 36.7 | 42.3 | 49.6 | µs |
| $t_{LOW}$ | SCL Low Period | 36.7 | 42.3 | 49.6 | µs |
| $t_{HD:DAT}$ Data hold time. | Master Transmit | 18.2 | 21.1 | 25 | µs |
| | Master Receive | 0 | | | ns |
| | Master Acknowledge | 300 | | | ns |
| $t_{SU:DAT}$ Data setup time. | Master Transmit | 18.2 | 21.1 | 25 | µs |
| | Master Receive | 250 | | | ns |
| | Master Acknowledge | 36.7 | 42.3 | 49.6 | µs |

| Master Operation (SMBSPEED bits = 11) | | | | | |
|---|---|---|---|---|---|
| $f_{SCL}$ | SCL clock frequency. | 6.16 | 7.17 | 8.19 | KHz |
| $t_{HD:STA}$ | Hold time after a (Repeated) Start condition. | 60.2 | 69.1 | 81 | µs |
| $t_{SU:STA}$ | Repeated Start setup time. | 149.7 | 171.4 | 200.3 | µs |
| $t_{SU:STO}$ | Stop condition setup time. | 60.2 | 69.1 | 81 | µs |
| $t_{HIGH}$ | SCL High Period | 60.7 | 69.7 | 81.6 | µs |
| $t_{LOW}$ | SCL Low Period | 60.7 | 69.7 | 81.6 | µs |
| $t_{HD:DAT}$ Data hold time. | Master Transmit | 30.2 | 34.9 | 41 | µs |
| | Master Receive | 0 | | | ns |
| | Master Acknowledge | 300 | | | ns |
| $t_{SU:DAT}$ Data setup time. | Master Transmit | 30.2 | 34.9 | 41 | µs |
| | Master Receive | 250 | | | ns |
| | Master Acknowledge | 60.7 | 69.7 | 81.6 | µs |

Table 3, SMBus AC Characteristics

BATT=12V, AGND=GND=CS-=0V, $V_{DD}$=$V_{AA}$, $\overline{SHDN}$ =3.4V, $C_{VAA}$=0.47 µF, INSTOSC=OFF, 32.768kHz on OSC2 **$T_A$ = 0°C to +85°C**, unless otherwise noted.

| SYMBOL | PARAMETER/CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| $V_{IL}$ | Clock/Data input low voltage. | 0.8 | | | V |
| $V_{IH}$ | Clock/Data input high voltage. | | | 2.1 | V |
| SCL and SDA Output Low Voltage | $I_{SINK}$ = 2mA | | | 0.4 | V |
| SCL and SDA Output High Leakage Current | $V_{SCL}$ = $V_{SDA}$ = 28V | | | 1 | µA |
| SCL and SDA short circuit current limit | $V_{SCL}$ = $V_{SDA}$ = 28V | 6 | | 50 | mA |

Table 4, SMBus DC Characteristics

# Electrical Characteristics (continued, $T_A$ = -40°C to +85°C)

BATT=12V, AGND=GND=CS=0V, VDD=VAA, $\overline{SHDN}$ =3.4V, $C_{VAA}$=0.47 µF, INSTOSC=OFF, 32.768kHz on OSC2 **$T_A$ = -40°C to +85°C**, unless otherwise noted.

| PARAMETER | CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| BATT LINEAR REGULATOR (VAA) | | | | | |
| Input Voltage Range | Includes dropout operation | 3.5 | | 28 | V |
| Output Voltage | 4V < BATT < 28V, 0 < $I_{LOAD}$ < 10mA | 3.2 | | 3.6 | V |
| Short Circuit Current | VAA = 0V, 2.5V < BATT < 28V | 15 | | 120 | mA |
| Power Fail Warning Trip Level (PFW\) | Falling VAA | 2.91 | | 3.09 | V |
| Power On Reset Trip Level ($\overline{MCLR}$) | Rising VAA | 2.62 | | 2.78 | V |
| SUPPLY CURRENT (BATT) | | | | | |
| SHUTDOWN: Everything Off | $\overline{SHDN}$ < 0.4V | | | 1 | µA |
| ON-CHIP TEMPERATURE SENSOR | | | | | |
| Temperature Measurement Error | | | +/-8 | | °K |
| OVERCHARGE AND OVERDISCHARGE COMPARATORS | | | | | |
| ODI, OCI Input Offset Voltage | | -7 | | 7 | mV |
| ODI, OCI Input Bias Current | FET input comparators | -1 | | 1 | µA |
| ODI, OCI Comparator Propagation Delay | 20mV input overdrive | | | 30 | µs |
| DIS, CHG Sink Current | DIS, CHG = 2V | 20 | | 80 | µA |
| DIS, CHG Sink Current | DIS, CHG = 20V, BATT = 20V | 30 | | 100 | µA |
| DIS, CHG Source Current | 0V < DIS, CHG < BATT-2V | 3 | | | mA |
| Leakage Current | DIS = CHG = 28V | | | 1 | µA |
| INSTRUCTION OSCILLATOR | | | | | |
| Frequency | -40°C to +85°C, and 2.8V < VDD < 3.6V | 3 | | 4 | MHz |
| 32KHZ TIMER OSCILLATOR | | | | | |
| OSC1 Input Current | OSC1=0V or 3.4V, OSC2 is floating | | | 500 | nA |
| OSC2 Sink Current | OSC1= OSC2 = 3.4V | 4 | | 20 | µA |
| OSC2 Source Current | OSC1= OSC2=0V | 3 | | 20 | µA |
| Transconductance | | 1.3 | | 12 | µA/V |
| LOGIC INPUTS/OUTPUTS | | | | | |
| IO[7:0] Input Voltage Low | | | | 0.8 | V |
| IO[7:0] Input Voltage High | | 2.4 | | | V |
| IO[7:0] High Impedance Leakage Current | I/O pins programmed to Hi-Z. | | | 1 | µA |
| IO[7:0] Output Voltage Low | $I_{SINK}$ = 2mA | | | 0.4 | V |
| IO[7:0] Output Voltage High | $I_{SOURCE}$ = 2mA | VDD-0.6 | | | V |

# Electrical Characteristics (continued, $T_A$ = -40°C to +85°C)

BATT=12V, AGND=GND=CS-=0V, VDD=VAA, $\overline{SHDN}$ =3.4V, $C_{VAA}$=0.47 µF, INSTOSC=OFF, 32.768kHz on OSC2 **$T_A$ = -40°C to +85°C**, unless otherwise noted.

| PARAMETER | CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| $\overline{MCLR}$ RESET PIN | | | | | |
| $\overline{MCLR}$ Output Voltage Low | $I_{SINK}$ = 1mA, and BATT = VAA = VDD = 2.5V | | | 0.4 | V |
| $\overline{MCLR}$ Output Voltage High | $I_{SOURCE}$ = 0 BATT = VAA = VDD = 3.4V | VDD - 0.2 | | | V |
| $\overline{MCLR}$ Input High Voltage | | 2.4 | | | V |
| $\overline{MCLR}$ Input Low Voltage | | | | 1 | V |
| SHUTDOWN PIN LOGIC LEVELS ($\overline{SHDN}$ ) | | | | | |
| Input High Voltage | | 2.2 | | | V |
| Input Low Voltage | | | | 0.4 | V |
| Input Bias Current | $\overline{SHDN}$ = 3.4V | | | 3.0 | µA |
| | $\overline{SHDN}$ = 28V | | | 60 | µA |
| HIGH-VOLTAGE OPEN DRAIN OUTPUTS THAT PULL TO GND (HV0-HV5) | | | | | |
| Output Voltage Low | $I_{SINK}$ = 1 mA | | | 0.4 | V |
| Leakage Current | Output High, $V_{APPLIED}$ = 28V | | | 1 | µA |
| HIGH-VOLTAGE OPEN COLLECTOR OUTPUTS THAT PULL TO BATT (HV6 and HV7) | | | | | |
| Output Voltage High | $I_{SOURCE}$ = 100µA | BATT-0.5 | | | V |
| Leakage Current | Output Low $V_{APPLIED}$ = 0V | | | 1 | µA |

## SPI Interface Electrical Characteristics ($T_A$ = -40°C to +85°C)

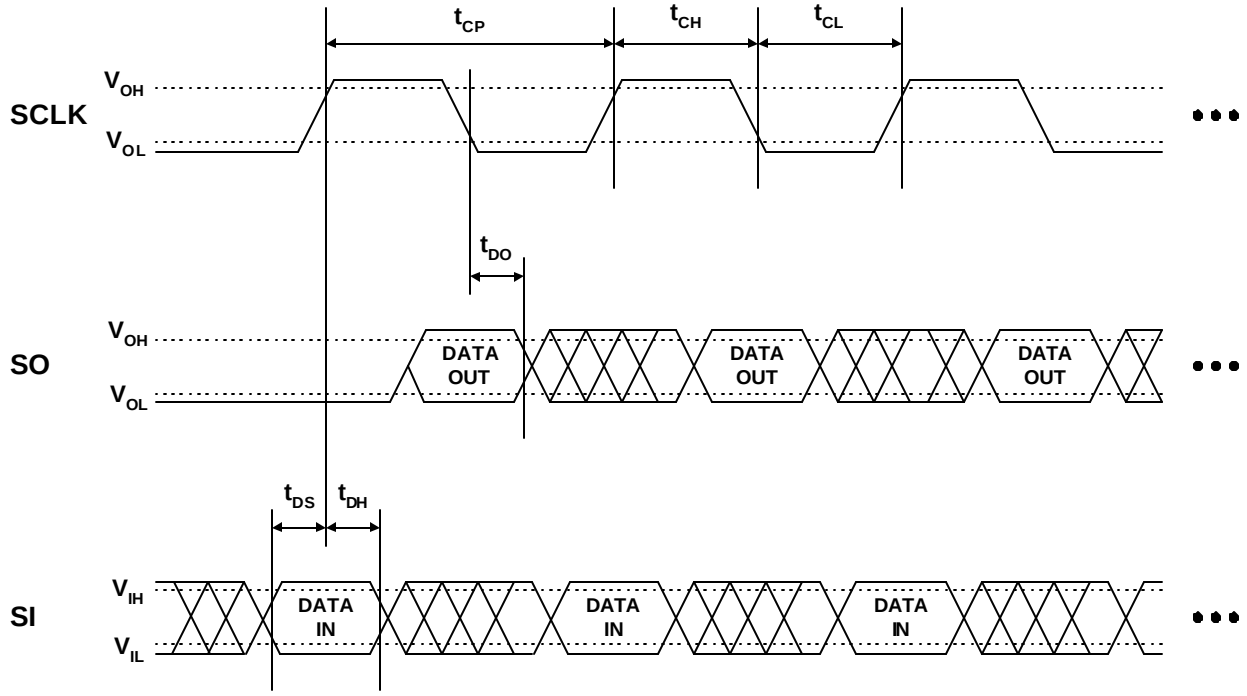



Figure 36, SPI Interface Timing ($T_A$ = -40°C to +85°C)

BATT=12V, AGND=GND=CS-=0V, VDD=VAA, $\overline{SHDN}$ =3.4V, $C_{VAA}$=0.47 µF, INSTOSC=OFF, 32.768kHz on OSC2 **$T_A$ = -40°C to +85°C**, unless otherwise noted.

| SYMBOL | PARAMETER/CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| SPI INTERFACE AC TIMING *($C_{LOAD}$ = 20pF)* | | | | | |
| $t_{CP}$ | SCLK Period | 250 | | | ns |
| $t_{CH}$ | SCLK Pulse Width High | 100 | | | ns |
| $t_{CL}$ | SCLK Pulse Width Low | 100 | | | ns |
| $t_{DO}$ | SCLK Fall to SO Valid | -30 | | 30 | ns |
| $t_{DS}$ | SI to SCLK Data Setup Time | 70 | | | ns |
| $t_{DH}$ | SI to SCLK Data Hold Time | 0 | | | ns |
| SPI INTERFACE DC CHARACTERISTICS | | | | | |
| $V_{OH}$ | SPI Output High; $I_{SOURCE}$ = 2mA | VDD-0.6 | | | V |
| $V_{OL}$ | SPI Output Low; $I_{SOURCE}$ = 2mA | | | 0.4 | V |
| $V_{IH}$ | SPI Input High | 2.4 | | | V |
| $V_{IL}$ | SPI Input Low | | | 0.8 | V |

# SMBus Interface Electrical Characteristics ($T_A$ = -40°C to +85°C)

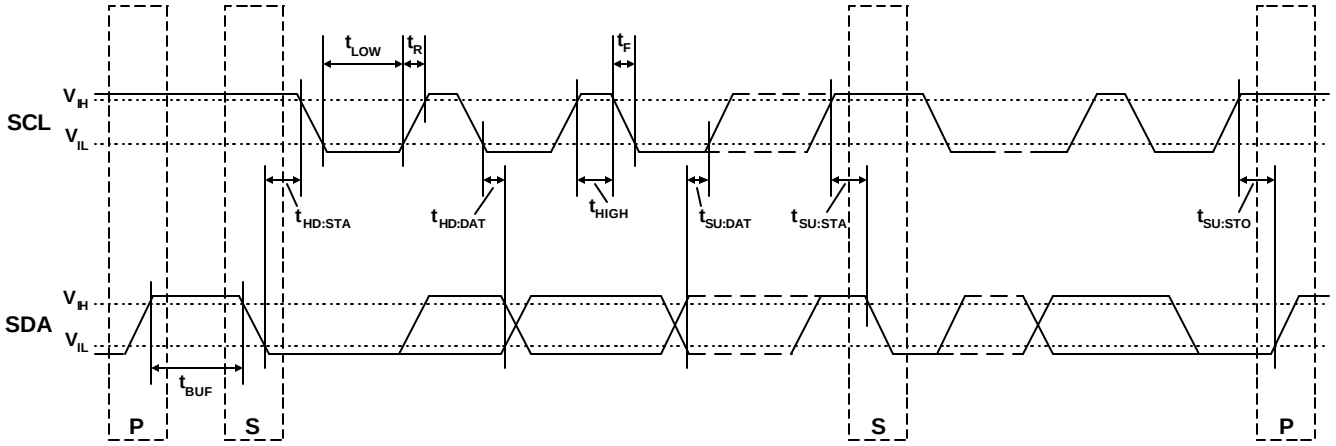



Figure 37, SMBus Timing Diagram ($T_A$ = -40°C to +85°C)

BATT=12V, AGND=GND=CS-=0V, VDD=VAA, $\overline{SHDN}$ =3.4V, $C_{VAA}$=0.47 µF, INSTOSC=OFF, 32.768kHz on OSC2 **$T_A$ = -40°C to +85°C**, unless otherwise noted.

| SYMBOL | PARAMETER/CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| Slave Operation | | | | | |
| $t_{BUF}$ | Bus free time between a Stop and Start condition. | 4.7 | | | µs |
| $t_{HD:STA}$ | Hold time after a (Repeated) Start condition. | 4.0 | | | µs |
| $t_{SU:STA}$ | Repeated Start setup time. | 4.7 | | | µs |
| $t_{SU:STO}$ | Stop condition setup time. | 4.0 | | | µs |
| $t_{HD:DAT}$ Data hold time. | Transmit | 300 | | | ns |
| | Receive | 0 | | | ns |
| $t_{SU:DAT}$ Data setup time. | Transmit | 250 | | | ns |
| | Receive | 250 | | | ns |
| $t_{TIMEOUT}$ | Detect clock (SCL) low timeout. | | | 35 | ms |
| $t_{LOW}$ | Clock (SCL) low period. | 4.7 | | | µs |
| $t_{HIGH}$ | Clock (SCL) high period. | 4.0 | | | µs |

BATT=12V, AGND=GND=CS-=0V, VDD=VAA, $\overline{SHDN}$ =3.4V, $C_{VAA}$=0.47 µF, INSTOSC=OFF, 32.768kHz on OSC2 **$T_A$ = -40°C to +85°C**, unless otherwise noted.

| SYMBOL | PARAMETER/CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| All Master Operations | | | | | |
| $t_{TIMEOUT}$ | Detect clock (SCL) low timeout. | | | 35 | ms |
| Master Operation (SMBSPEED bits = 00) | | | | | |
| $f_{SCL}$ | SCL clock frequency. | 42.2 | | 56.5 | KHz |
| $t_{HD:STA}$ | Hold time after a (Repeated) Start condition. | 8.2 | | 11.6 | µs |
| $t_{SU:STA}$ | Repeated Start setup time. | 19.7 | | 27 | µs |
| $t_{SU:STO}$ | Stop condition setup time. | 8.2 | | 11.6 | µs |
| $t_{HIGH}$ | SCL High Period | 8.7 | | 12.3 | µs |
| $t_{LOW}$ | SCL Low Period | 8.7 | | 12.3 | µs |
| $t_{HD:DAT}$ Data hold time. | Master Transmit | 4.2 | | 6.3 | µs |
| | Master Receive | 0 | | | ns |
| | Master Acknowledge | 300 | | | ns |
| $t_{SU:DAT}$ Data setup time. | Master Transmit | 4.2 | | 6.3 | µs |
| | Master Receive | 250 | | | ns |
| | Master Acknowledge | 8.7 | | 12.3 | µs |
| Master Operation (SMBSPEED bits = 01) | | | | | |
| $f_{SCL}$ | SCL clock frequency. | 17.96 | | 23.81 | KHz |
| $t_{HD:STA}$ | Hold time after a (Repeated) Start condition. | 20.2 | | 27.6 | µs |
| $t_{SU:STA}$ | Repeated Start setup time. | 49.7 | | 67 | µs |
| $t_{SU:STO}$ | Stop condition setup time. | 20.2 | | 27.6 | µs |
| $t_{HIGH}$ | SCL High Period | 20.7 | | 28.3 | µs |
| $t_{LOW}$ | SCL Low Period | 20.7 | | 28.3 | µs |
| $t_{HD:DAT}$ Data hold time. | Master Transmit | 10.2 | | 14.3 | µs |
| | Master Receive | 0 | | | ns |
| | Master Acknowledge | 300 | | | ns |
| $t_{SU:DAT}$ Data setup time. | Master Transmit | 10.2 | | 14.3 | µs |
| | Master Receive | 250 | | | ns |
| | Master Acknowledge | 20.7 | | 28.3 | µs |
| Master Operation (SMBSPEED bits = 10) | | | | | |
| $f_{SCL}$ | SCL clock frequency. | 10.17 | | 13.56 | KHz |
| $t_{HD:STA}$ | Hold time after a (Repeated) Start condition. | 36.2 | | 49 | µs |
| $t_{SU:STA}$ | Repeated Start setup time. | 89.7 | | 120.3 | µs |
| $t_{SU:STO}$ | Stop condition setup time. | 36.2 | | 49 | µs |
| $t_{HIGH}$ | SCL High Period | 36.7 | | 49.6 | µs |
| $t_{LOW}$ | SCL Low Period | 36.7 | | 49.6 | µs |
| $t_{HD:DAT}$ Data hold time. | Master Transmit | 18.2 | | 25 | µs |
| | Master Receive | 0 | | | ns |
| | Master Acknowledge | 300 | | | ns |
| $t_{SU:DAT}$ Data setup time. | Master Transmit | 18.2 | | 25 | µs |
| | Master Receive | 250 | | | ns |
| | Master Acknowledge | 36.7 | | 49.6 | µs |

| Master Operation (SMBSPEED bits = 11) | | | | | |
|---|---|---|---|---|---|
| $f_{SCL}$ | SCL clock frequency. | 6.16 | | 8.19 | KHz |
| $t_{HD:STA}$ | Hold time after a (Repeated) Start condition. | 60.2 | | 81 | µs |
| $t_{SU:STA}$ | Repeated Start setup time. | 149.7 | | 200.3 | µs |
| $t_{SU:STO}$ | Stop condition setup time. | 60.2 | | 81 | µs |
| $t_{HIGH}$ | SCL High Period | 60.7 | | 81.6 | µs |
| $t_{LOW}$ | SCL Low Period | 60.7 | | 81.6 | µs |
| $t_{HD:DAT}$ Data hold time. | Master Transmit | 30.2 | | 41 | µs |
| | Master Receive | 0 | | | ns |
| | Master Acknowledge | 300 | | | ns |
| $t_{SU:DAT}$ Data setup time. | Master Transmit | 30.2 | | 41 | µs |
| | Master Receive | 250 | | | ns |
| | Master Acknowledge | 60.7 | | 81.6 | µs |

Table 5, SMBus AC Characteristics (**$T_A$ = -40°C to +85°C**)

BATT=12V, AGND=GND=CS-=0V, VDD=VAA, $\overline{SHDN}$ =3.4V, $C_{VAA}$=0.47 µF, INSTOSC=OFF, 32.768kHz on OSC2 **$T_A$ = -40°C to +85°C**, unless otherwise noted.

| SYMBOL | PARAMETER/CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| $V_{IL}$ | Clock/Data input low voltage. | 0.8 | | | V |
| $V_{IH}$ | Clock/Data input high voltage. | | | 2.1 | V |
| SCL and SDA Output Low Voltage | $I_{SINK}$ = 2mA | | | 0.4 | V |
| SCL and SDA Output High Leakage Current | $V_{SCL}$ = $V_{SDA}$ = 28V | | | 1 | µA |
| SCL and SDA short circuit current limit | $V_{SCL}$ = $V_{SDA}$ = 28V | 6 | | 50 | mA |

Table 6, SMBus DC Characteristics (**$T_A$ = -40°C to +85°C**)

## Typical Operating Characteristics

(TA = +25°C, unless otherwise noted)

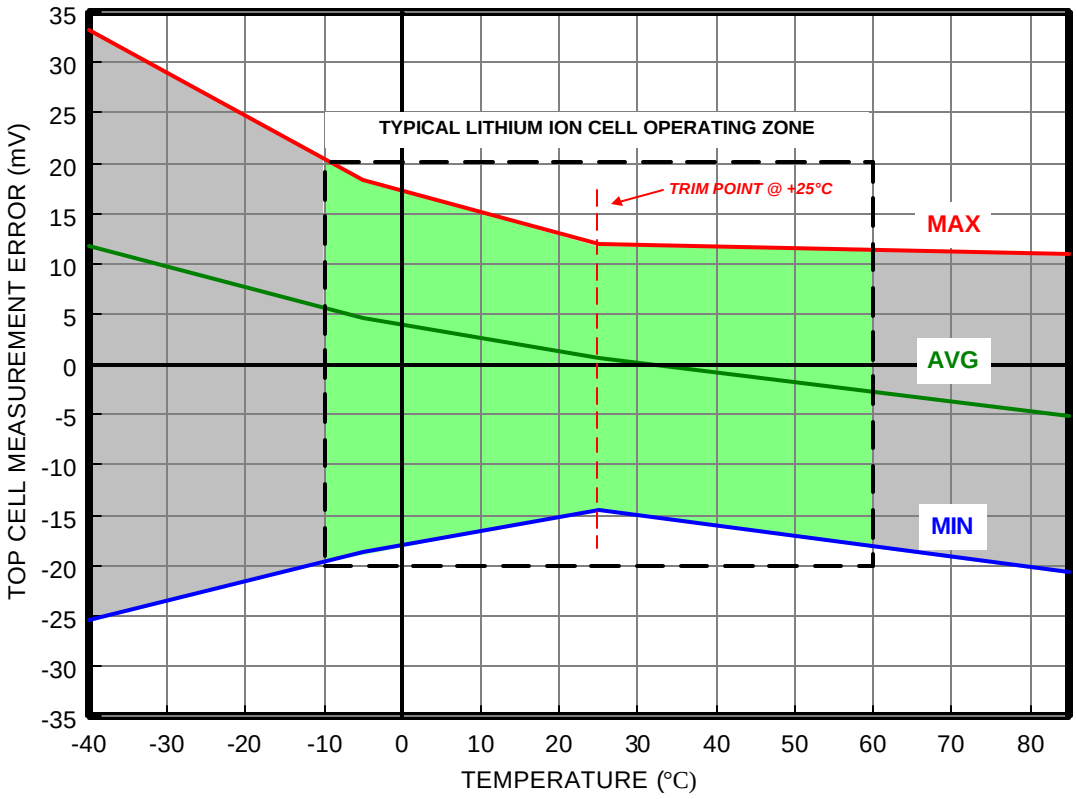



Figure 38, Typical ADC Voltage Measurement Error

## Typical Operating Characteristics (continued)

(TA = +25°C, unless otherwise noted)

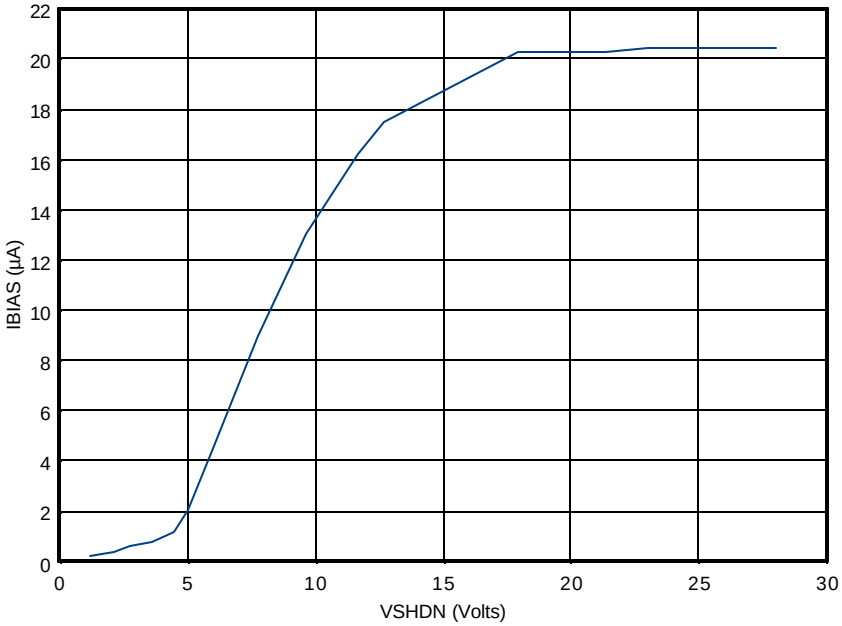



Figure 39, $\overline{SHDN}$ Input Bias Current vs. $V_{SHDN}$

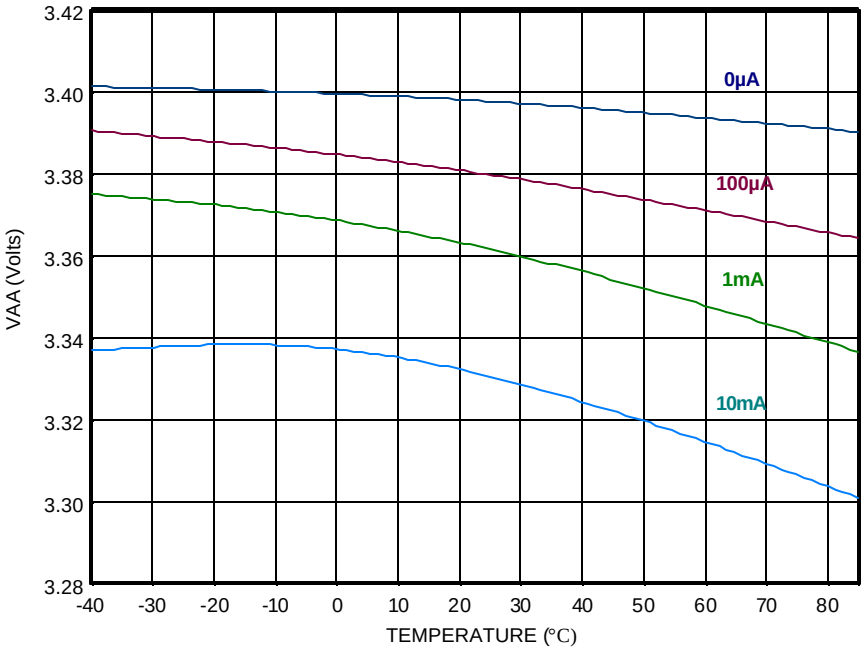



Figure 40, $V_{AA}$ Output Voltage vs. Load Current and Temperature

## Typical Operating Characteristics (continued)

(TA = +25°C, unless otherwise noted)

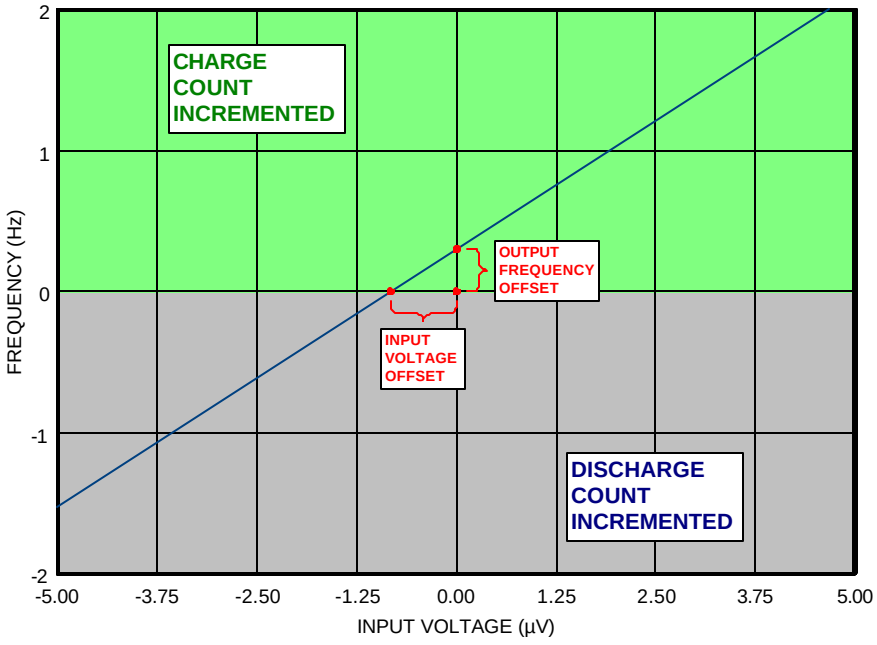



Figure 41, Fuel Gauge Frequency vs. Input Voltages Near Zero

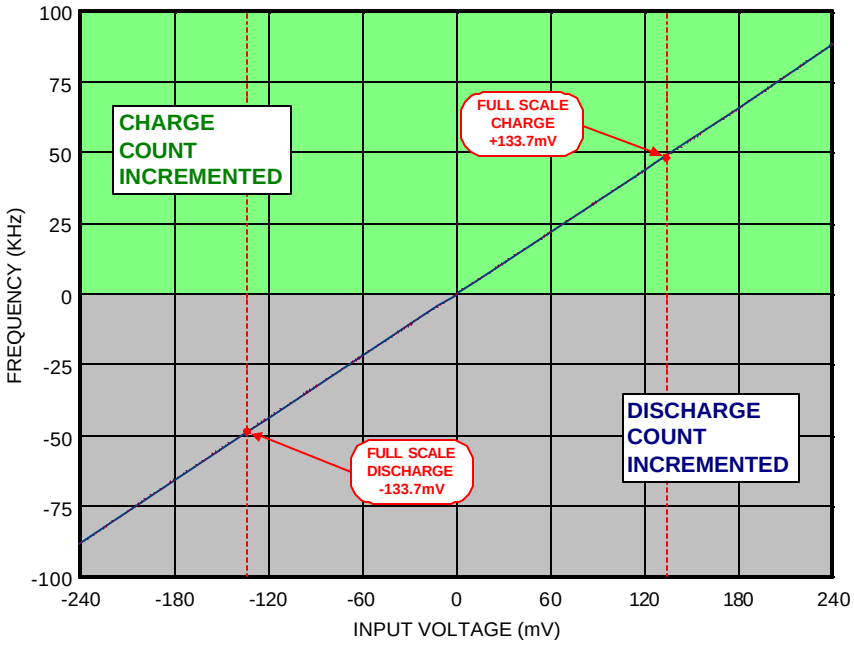


Figure 42, Fuel Gauge Frequency vs. Input Voltage

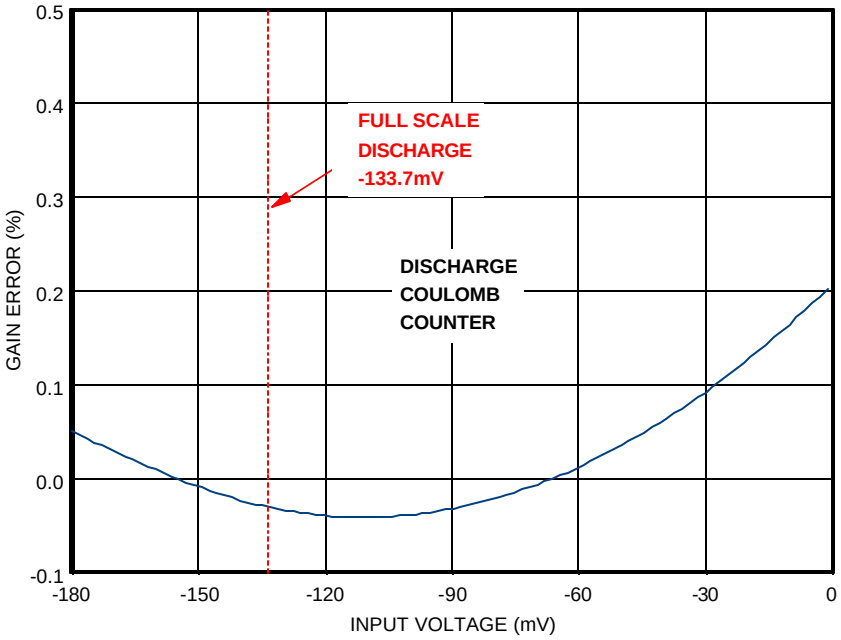



Figure 43, Discharge Gain Error vs. Fuel Gauge Input Voltage

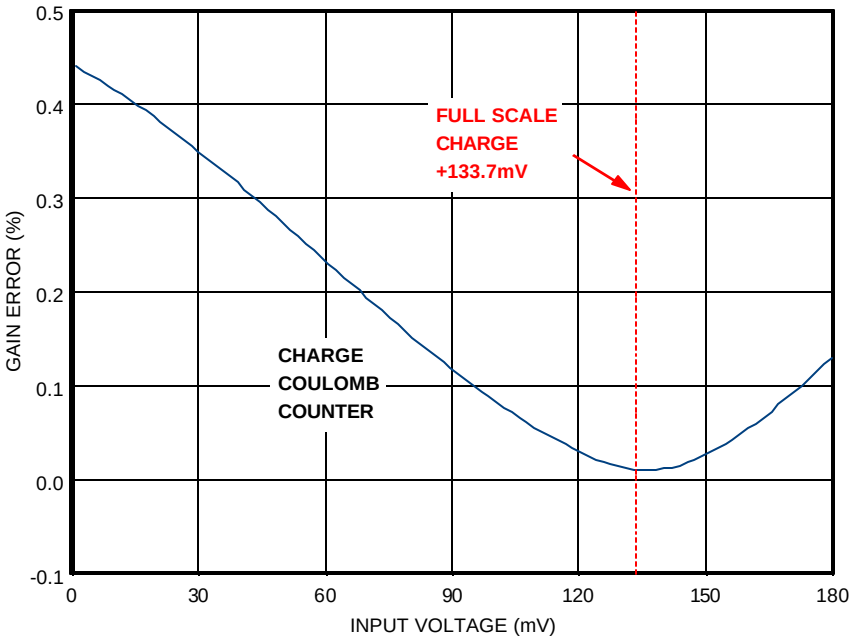
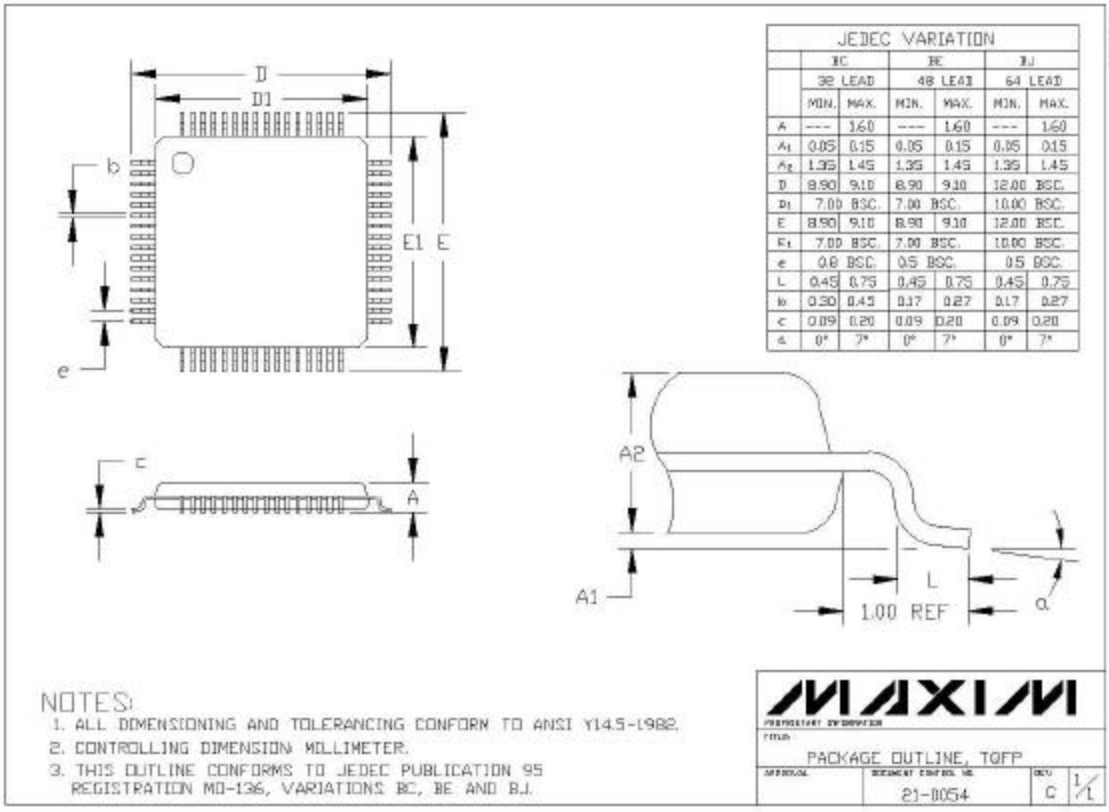



Figure 44, Charge Gain Error vs. Fuel Gauge Input Voltage

# Package Information




| JEDEC VARIATION | | | | | | |
|---|---|---|---|---|---|---|
| | BC | | BE | | BJ | |
| | 32 LEAD | | 48 LEAD | | 64 LEAD | |
| | MIN. | MAX. | MIN. | MAX. | MIN. | MAX. |
| A | --- | 1.60 | --- | 1.60 | --- | 1.60 |
| $A_1$ | 0.05 | 0.15 | 0.05 | 0.15 | 0.05 | 0.15 |
| $A_2$ | 1.35 | 1.45 | 1.35 | 1.45 | 1.35 | 1.45 |
| D | 8.90 | 9.10 | 8.90 | 9.10 | 12.00 BSC. | |
| $D_1$ | 7.00 BSC. | | 7.00 BSC. | | 10.00 BSC. | |
| E | 8.90 | 9.10 | 8.90 | 9.10 | 12.00 BSC. | |
| $E_1$ | 7.00 BSC. | | 7.00 BSC. | | 10.00 BSC. | |
| e | 0.8 BSC. | | 0.5 BSC. | | 0.5 BSC. | |
| L | 0.45 | 0.75 | 0.45 | 0.75 | 0.45 | 0.75 |
| b | 0.30 | 0.45 | 0.17 | 0.27 | 0.17 | 0.27 |
| c | 0.09 | 0.20 | 0.09 | 0.20 | 0.09 | 0.20 |
| α | 0° | 7° | 0° | 7° | 0° | 7° |

NOTES:
1. ALL DIMENSIONING AND TOLERANCING CONFORM TO ANSI Y14.5-1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. THIS OUTLINE CONFORMS TO JEDEC PUBLICATION 95 REGISTRATION MO-136, VARIATIONS BC, BE AND BJ.

MAXIM

PACKAGE OUTLINE, TQFP

21-0054 C 1/1

# Appendix

## An Overview Of Smart Batteries

Over the past five years, Smart Batteries, incorporating Lithium Ion cells, have evolved as the industry standard for supplying power to notebook computers. Smart Batteries vary widely in their exact composition, however common to all is their integration of rechargeable cells capable of providing power, with electronic measurement and control circuitry. Some Smart Batteries, depending on the cell chemistry used, also have electronic circuitry to protect the battery cells from being destroyed by either the battery charger or the notebook computer.

### The Smart Battery In A Notebook Power Supply System

Figure 45 illustrates a typical notebook computer power supply system. Note that the Smart Battery can communicate with other devices, such as the Host Computer, or Smart Battery Charger, via two separate communication interfaces. The primary communication interface, the System Management Bus (SMBus), is a two wire, bi-directional serial bus that provides a simple, efficient interface for data exchange between devices. The SMBus uses a serial data line (SDA) and a serial clock line (SCL) for data transfer. Using the SMBus, the Smart Battery can provide data when requested, send charging information to a charger, and broadcast critical alarm information when parameters (measured or calculated) exceed predetermined limits within the particular battery system.

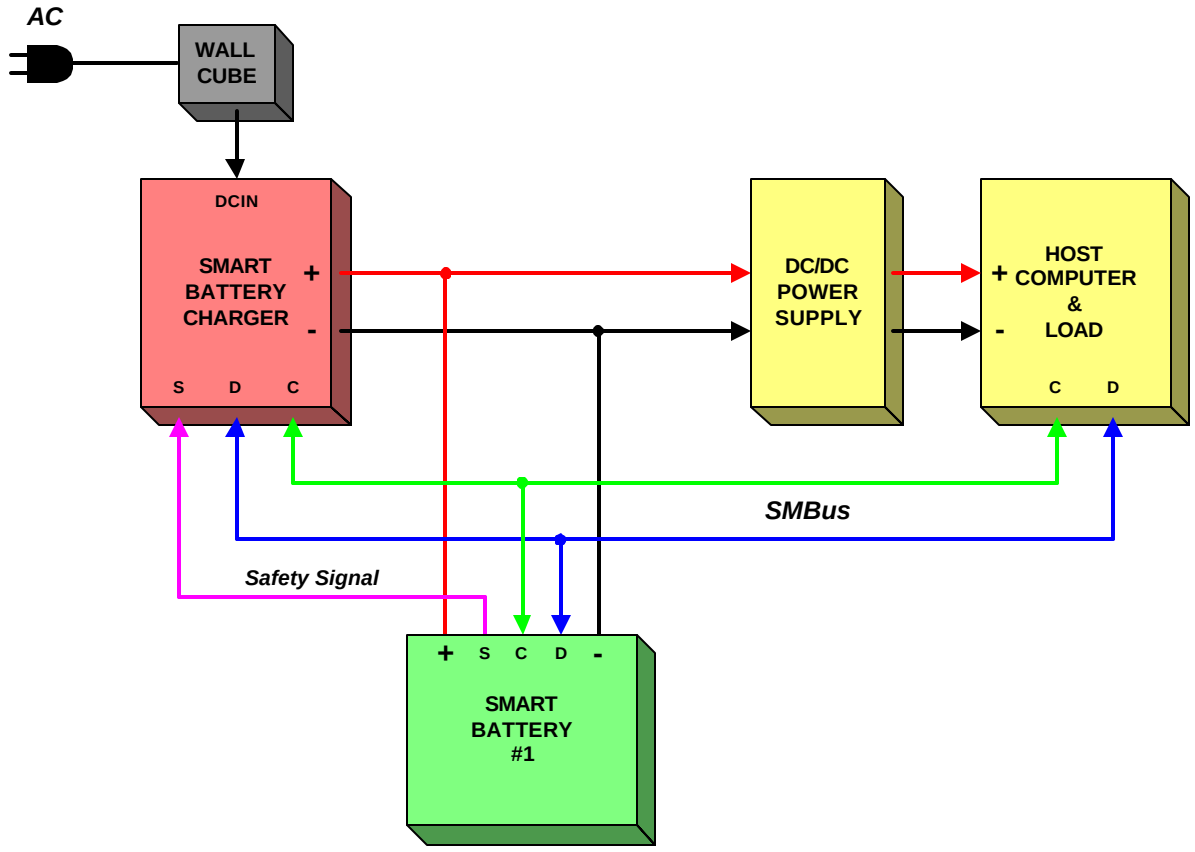



Figure 45, Simplified Notebook Computer Power Supply System

In the power supply system described by Figure 45, the Smart Battery can be both a Slave and a Master SMBus device. When responding to requests from the Host computer, it is a Slave device. When it broadcasts charge current and voltage requirements to the Smart Charger, it is an SMBus Master device. Note that there is a single-line Safety Signal that is connected between the Smart Battery and the Smart Charger. It communicates the Smart

Battery's gross temperature (COLD, NORMAL, and HOT), and the Smart Charger then terminates or inhibits charge, depending on the temperature information sent. The Safety Signal is also an alternate signaling method should the SMBus interface become inoperable. Battery chargers often use the "S" Pin to confirm correct charging.

## What Makes Smart Batteries "Smart"?

Most Smart or Intelligent Battery systems not only calculate real-time parametric data, they also predict battery performance based on given load conditions. The ability to provide predictive performance information makes them truly smart. The Smart Battery also stores portions of measured data to maintain a historical record of operation. This historical battery data is stored in non-volatile memory device such as an EEPROM. By storing this historical data in EEPROM, the Smart Battery can maintain its capacity information even when being moved from one notebook computer to another.

## Lithium Ion Cell Protection

In Smart Battery packs designed with Lithium Ion cells, there are generally protection circuits to prevent cell voltages from exceeding manufacturer recommended limits or excessive currents caused by over charging, and shorted cells.

Figure 46 shows a Smart Battery pack implemented with the MAX1780 Advanced Smart Battery Pack Controller.

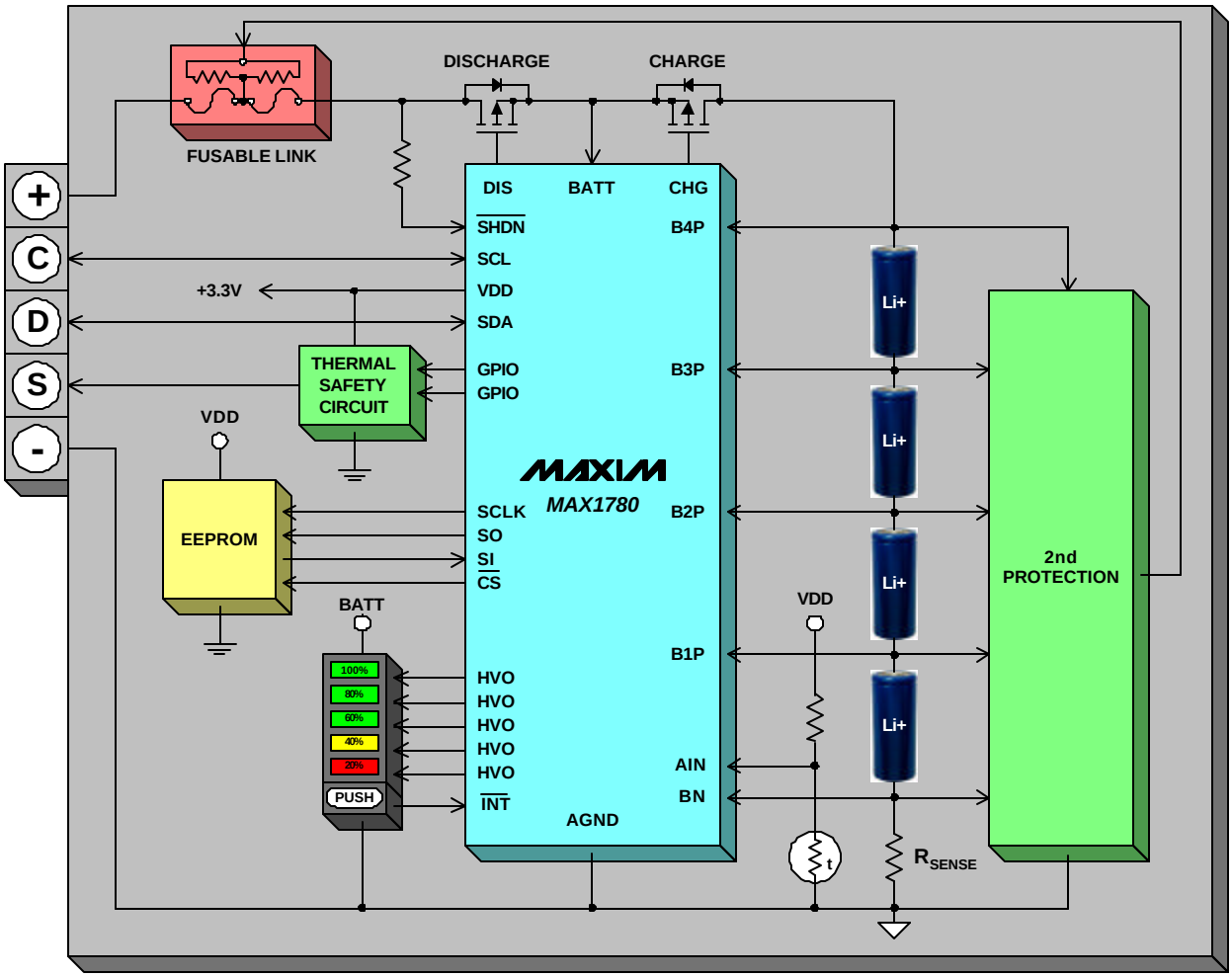



**Figure 46, Typical Smart Battery Pack Implemented With The MAX1780**

## Instruction Set Summary

Each MAX1780 instruction is a 12-bit word comprised of an Opcode, which specifies the instruction type, and one or more operands that further specify the operation of the instruction.

For **byte-oriented** instructions, **"f"** represents a file register designator and **"d"** represents a destination designator. The file register designator is used to specify which one of the file registers is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If **"d"** is '0', the result is placed in the "W" Register. If **"d"** is '1', the result is placed in the file register specified in the instruction.

For **bit-oriented** instructions, **"b"** represents a bit field designator that selects the number of the bit affected by the operation, while **"f"** represents the number of the file in which the bit is located.

For **literal and control** operations, **"k"** represents an 8 or 9-bit constant or literal value.

All instructions are executed within one single instruction cycle, unless a conditional test is true or the program counter is changed as a result of an instruction. In this case, the execution takes two instruction cycles. One instruction cycle consists of four instruction oscillator clocks. Therefore, for a nominal instruction oscillator frequency of 3.5 MHz, the normal instruction execution time is approximately 1.14 µs.

| Mnemonic | Description | Cycles | Encoding | Status | Notes |
|---|---|---|---|---|---|
| ADDWF f, d | Adds the contents of the "W" register to contents of selected register. Results in "W" or f. | 1 | 0001 11df ffff | C, DC, Z | |
| ANDLW k | ANDs the contents of the "W" register with literal (k) contained in instruction. Result in "W". | 1 | 1110 kkkk kkkk | Z | |
| ANDWF f, d | ANDs the contents of the "W" register with contents of selected register. Result in "W" or f. | 1 | 0001 01df ffff | Z | |
| BCF f, b | Clears selected bit in selected register to 0. | 1 | 0100 bbbf ffff | None | |
| BSF f, b | Sets selected bit in selected register to 1. | 1 | 0101 bbbf ffff | None | |
| BTFSC f, b | Tests specified bit in selected register. Skips the next instruction if bit tested is clear (0). | 1(2) | 0110 bbbf ffff | None | |
| BTFSS f, b | Tests specified bit in selected register. Skips the next instruction if bit tested is set (1). | 1(2) | 0111 bbbf ffff | None | |
| CALL k | Call subroutine at specified starting address (k). | 2 | 1001 kkkk kkkk | None | |
| CLRF f | Clears selected register to 0. | 1 | 0000 011f ffff | Z | |
| CLRW | Clears "W" register to 0. | 1 | 0000 0100 0000 | Z | |
| CLRWDT | Clear Watchdog Timer (reset to 0). Also resets the WDT prescaler. | 1 | 0000 0000 0100 | None | |
| CLRTI | Clear Watchdog Timer interrupt latch(reset to 0). | 1 | 0000 0001 0000 | None | |
| COMF f, d | Complements selected register's contents (1's to 0's/0's to 1's). Result in "W" or f. | 1 | 0010 01df ffff | Z | |
| DECF f, d | Decrements the selected register. Decrementing 0x00 results in 0xff. Result in "W" or f. | 1 | 0000 11df ffff | Z | |
| DECFSZ f, d | Decrements specified register. Skips next instruction if register contents = 0. Result in "W" or f. | 1(2) | 0010 11df ffff | None | |
| FREE | The FREE instruction is used to write to the Program RAM and to the Interrupt Configuration Register. The instruction's specific action depends on the contents of FSR bits 5 and 6. | 1 | 0000 0000 0001 | None | |
| GOTO k | Go to specified address (k). | 2 | 101k kkkk kkkk | None | |
| INCF f, d | Increments the selected register. Incrementing 0xff results in 0x00. Result in "W" or f. | 1 | 0010 10df ffff | Z | |
| INCFSZ f, d | Increments specified register. Skips next instruction if register contents = 0. Result in "W" or f. | 1(2) | 0011 11df ffff | None | |
| IORLW k | OR's contents of the "W" register with literal (k) contained in instruction. Result in "W". | 1 | 1101 kkkk kkkk | Z | |
| IORWF f, d | OR's contents of the "W" register with the contents of selected register. Result in "W" or f. | 1 | 0001 00df ffff | Z | |
| MOVF f, d | Moves a copy of the selected register's contents into "W" or f. | 1 | 0010 00df ffff | Z | |

| | | | | | |
|---|---|---|---|---|---|
| MOVLW k | Loads the "W" register with literal (k). | 1 | 1100 kkkk kkkk | None | |
| MOVWF f | Moves a copy of the "W" register's contents into selected register. | 1 | 0000 001f ffff | None | |
| NOP | Do nothing for one instruction cycle. | 1 | 0000 0000 0000 | None | |
| OPTION | Load OPTION register with the contents of the "W" register. | 1 | 0000 0000 0010 | None | |
| RETFIE | Return from Interrupt. The "W" register is unaffected and INTOFF flag bit is cleared. | 2 | 0000 0000 1001 | INTOFF | |
| RETLW k | Return from subroutine. Load the "W" register with literal (k). | 2 | 1000 kkkk kkkk | None | |
| RETURN | Return from subroutine. The "W" register and INTOFF flag bit are unaffected. | 2 | 0000 0000 1000 | None | |
| RLF f, d | Rotates bits in selected register one position to the left. Bits rotate through the CARRY flag. Result in "W" or f. | 1 | 0011 01df ffff | C | |
| RRF f, d | Rotates bits in selected register one position to the right. Bits rotate through the CARRY flag. Result in "W" or f. | 1 | 0011 00df ffff | C | |
| SLEEP | Shuts down µC core (instruction oscillator) to reduce power consumption. Wake up via Reset, Watchdog Timer, or an external Interrupt. | 1 | 0000 0000 0011 | INTOFF, INTWDT | |
| SUBWF f, d | Subtracts the contents of the "W" register from the contents of selected register by 2's complement arithmetic. Results in "W" or f. | 1 | 0000 10df ffff | C, DC, Z | |
| SWAPF f, d | Exchanges the upper and lower nibbles (4 bits) of the selected register. Result in "W" or f. | 1 | 0011 10df ffff | None | |
| TRIS f | Loads the selected Port (A, B, or C) register with the contents of the "W" register. | 1 | 0000 0000 0fff | None | |
| XORLW k | XORs the contents of the "W" register with literal (k) contained in instruction. Result in "W". | 1 | 1111 kkkk kkkk | Z | |
| XORWF f, d | XORs the contents of the "W" register with the contents of selected register. Result in "W" or f. | 1 | 0001 10df ffff | Z | |

Legend:
b = Bit Address
d = Destination with "0" to "W" Register and "1" to File Register
f = 5-bit File Register Address
k = 8-bit Literal Value

# Errata

Although all the issues listed here are expected to be addressed in future revisions of the MAX1780, care should be used to evaluate the implications of these issues in any specific design.

The MAX1780 parts you have received conform functionally to this data sheet, except for the following issues.

## 1. ODO/OCO Section:

When the MAX1780 comes out of POR, the charge and discharge MOSFET drivers are asserted low.

**Impact:**

Cells are connected to the load before their voltages can be determined. In normal operation the MAX1780 will load program code from an external E2PROM after POR, and begin measuring the individual cell voltages within a few milliseconds. If the MAX1780 is unable to communicate with the external E2PROM, the voltage measurements will not be made. In such a case, the battery pack will then have to rely on an external second level protection element for safeguard against over voltage on cells.

**Solution/Workaround:**

Hardware change is required.

## 2. ODI/OCI Comparators:

When turning the protection MOSFETS ON (powering up the ODI/OCI comparators) in software, the comparators don't power up in a controlled manner.

**Impact:**

Spurious overcharge and discharge interrupts can occur.

**Solution/Workaround:**

A software workaround is currently used to minimize the effects of the problem. See the "Using Software To Control The Protection MOSFETs" section for an overview of the software workaround.

## 3. SPI Interface:

Over the full range of temperature and voltage, the MAX1780 may clock data into the SI pin up to 40ns too early for a 5MHz serial EEPROM.

**Impact:**

The MAX1780 does not meet SI setup and hold timing requirements for a 5MHz serial EEPROM.

**Solution/Workaround:**

A hardware change is required to guarantee SPI communications at 5MHz. There is no timing problem when operating the SPI interface at the slower SCLK speed.

## 4. SMBus Interface:

The MAX1780 SMBus Master Interface may have some difficulty communicating in a multi-master environment. There are three areas of concern:

a. The MAX1780 does not implement the Tlow:sext and Tlow:mext timeouts.

b. The MAX1780 does not perform SCL clock synchronization.

c. The MAX1780 does not do bit-by-bit data arbitration.

**Impact:**

Master communications in a multi-master environment may be unreliable.

**Solution/Workaround:**

Hardware changes are required to alleviate all three problems, however it is possible to implement the Tlow:sext and Tlow:mext timeouts in software. Byte-by-byte data arbitration can also be performed in software.