FS3861
Data Sheet

Intelligent Charger Management Controller

Rev. 1.0

Dec. 2004

Taipei Office:

28F, No.27, Sec. 2, Zhongzheng E. Rd.,
Danshui Town, Taipei County 251, Taiwan
Tel.：886-2-28094742
Fax：886-2-28094874
Web Site: http://www.fsc.com.tw

Hsinchu Office:

4F-5, No. 30, Taiyuan St., Zhubei City,
Hsinchu County 302, Taiwan
Tel.：886-3-5525296
Fax：886-3-5525946

Shenzhen Technical Support:

14F, Dingxin Building West, No. 1, Liuxian Blvd.,
Xili Town, Nanshan District
Shenzhen, Guangdong 518055,
P. R. China
Tel. :86-755-26528742
Fax :86-755-26528940

Table of Contents

1. General Description

The FS3861 is a low-cost high-performance Li+ single-cell 4.2v/4.1v battery charger control IC which includes all the required constant-current and constant-voltage regulations of charge functions addressed for linear charger mode operations in typical four phases: pre-charging conditioning, constant current, constant voltage, and charge terminations (usually based on the minimum current reached). The maintenance re-charge (or called post-charge stage) proceeds if the full-charged battery voltage is once again lower than the desired full-capacity voltage because of consumptions of its capacity which occurs either at the battery's internal voltage drop across its terminals, or at the use of the battery.

This chip with built-in 8-bit RISC-type MCU with 1K-word OTP PROM and 64-Byte data RAM employs a minimum numbers of external transistor and passive resistor & capacitor devices to fulfill complete charger implementations at cost-effective solutions.

The available 16-pin SSOP-16 package is offered for balanced area and cost effective requirements for size-sensitive applications.

The FS3861 is suitable for the control of charge sequences of a variety of portable battery-powered applications, such as cellular phone's travel and base charger devices, digital camera, digital-video camcorder (DV), MP3 player ,etc.

2. Features

- **Ideal for the Li-ion/polymer Single-Cell 4.2v/4.1v charge control.**
- **Built-in 8-bit RISC-typed MCU with 1K-word OTP program ROM and 64-Byte data RAM.**
- **Integrated voltage and current regulation with programmable charge current.**
- **Supports typical Li+ battery's charge sequences such as pre-charge (trickle-mode charge), C-C (constant-current charge), C-V (constant-voltage charge), charge terminations, and re-charge operations.**
- **Batter than 1% charge voltage regulation accuracy.**
- **Charge operation can be monitored by the external host through the general I/O data bus.**
- **Features of the PWM voltage generation is complimentary to the provision of look-up voltage table for use at specific intermediate charge voltages or detected values for the comparator's function.**
- **2 LED output for charge status.**
- **Optional Temp and battery ID input through voltage sense input.**
- **Low-cost peripheral components of capacitor and resistor combinations for minimum BOM cost in manufacturing considerations.**
- **Development kit of LQFP-64 ICE evaluation (EV) board and reference charge program available for prototype design and facilitating debug use.**
- **SSOP-16 Package.**

3. Applications

- **Cellular phone external base or built-in charger**
- **MP3 player**
- **External charger through USB**
- **Digital still camera (DSC)**
- **Digital video camcorder (DV)**
- **Portable electronic device charger, etc.**

4. Ordering Information

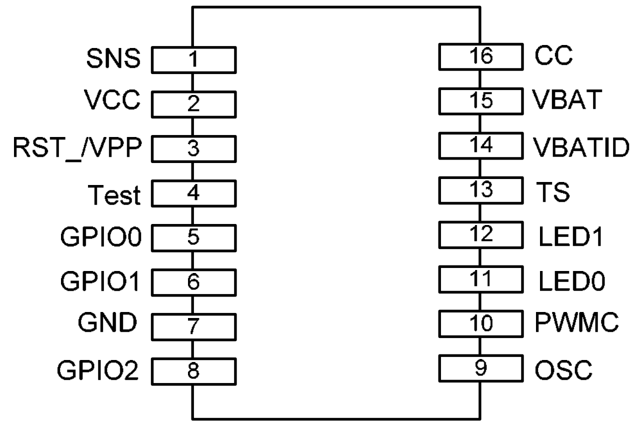
Product Number	Description	Package Type
FS3861-ICE	Customer can program the compiled hex code into EPROM through the FSC's development kit for evaluation and facilitating debug.	LQFP-64
FS3861A-nnnV	Customer's compiled hex code can be programmed by FSC or customer itself into EPROM at factory before shipping.	SSOP-16

Note1: Code number (nnnV) is assigned for customer.

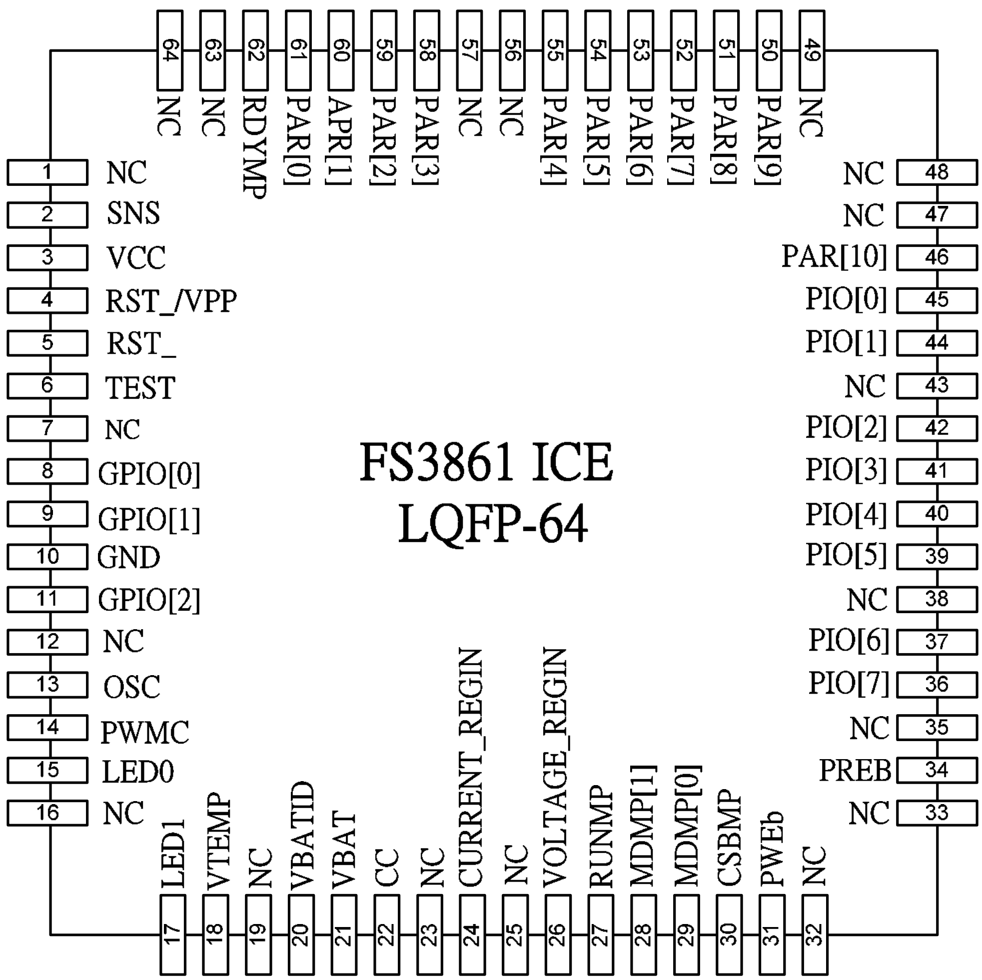
Note2: Code number (nnn = 001~999); Version (V = A~Z).

5. Pin Configuration

FS3861 SSOP16 Package



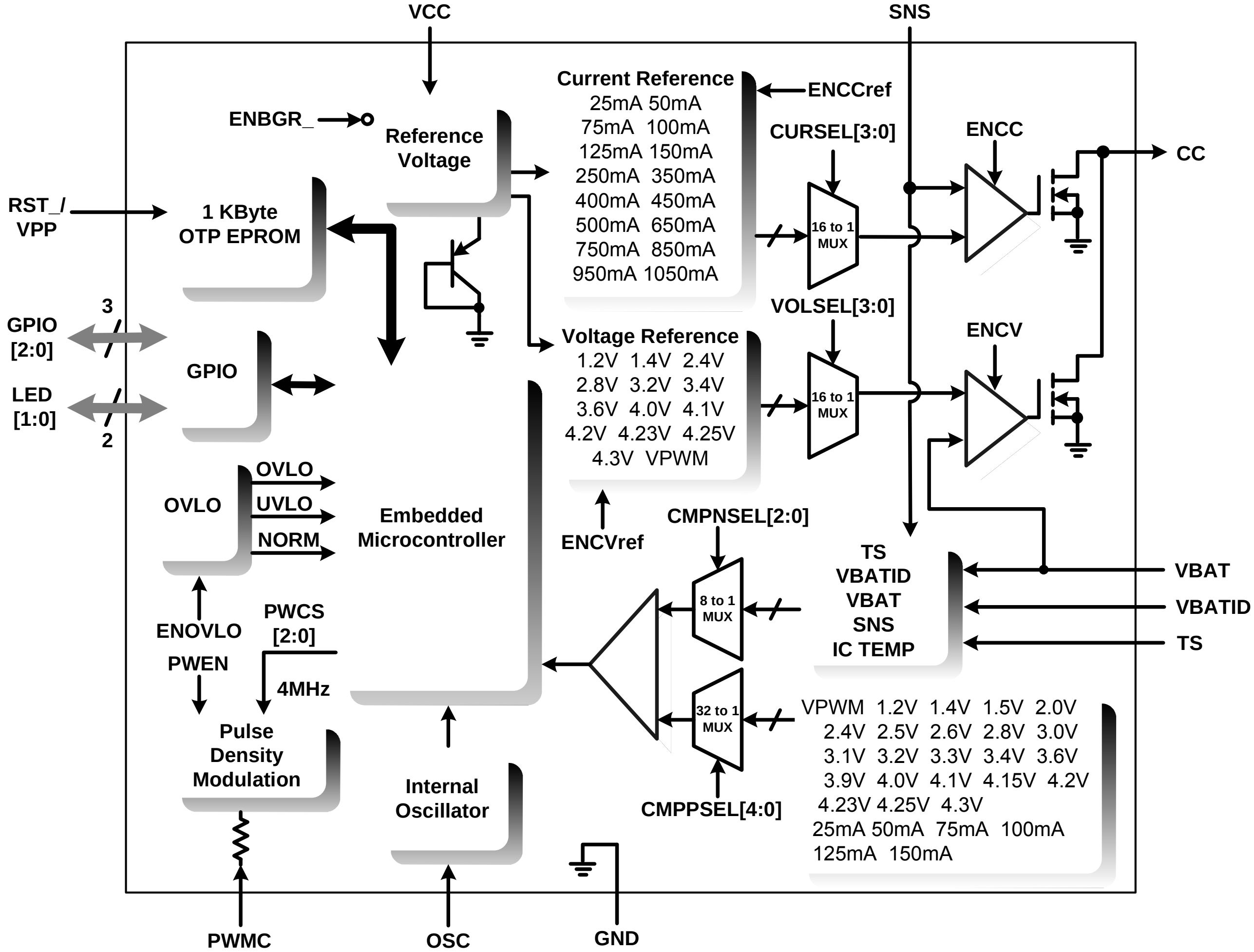
FS3861 ICE LQFP64 Package



6. Pin Description

Name	I/O	Pin No	Description
SNS	I	1	Current sensing using an external sensing resistor R_{SNS}
VCC	I	2	Supply voltage
RST_/VPP	I	3	Active low reset or as active high OTP program write
TEST	I	4	Test mode input. Test=1 is the normal mode. Test Mode is initiated while Test=0 before reset. This pin is suggested pulled inactive high for regular operation without Test Mode.
GPIO[0]	I/O	5	General purpose bi-directional I/O pin 0
GPIO[1]	I/O	6	General purpose bi-directional I/O pin 1
GND		7	Ground
GPIO[2]	I/O	8	General purpose bi-directional I/O pin 2
OSC	I	9	Oscillator input. Connect to an external resistor R=200kΩ, the oscillator frequency is around 4.5MHz
PWMC	I	10	PWM capacitor input for selection of the RC time constant in generating voltage reference.
LED0	O	11	Source or sink LED0 display
LED1	O	12	Source or sink LED1 display
TS	I	13	Battery temperature sensing input
VBATID	I	14	Battery ID-type selected by the voltage drop across the series resistor. Battery ID is for identification of either thick, thin battery or other selected types
VBAT	I	15	Battery input voltage
CC	O	16	Charge control output to drive pass transistor

7. Functional Block Diagram



8. Absolute Maximum Ratings

Parameter	Item	Rating	Unit
Supply Voltage to Ground Potential	VCC	-0.3 to 5.5	V
Applied Input Voltage for Programming OTP EPROM	VPP	-0.3 to 13	V
Applied input voltage of other pins	VIO	-0.3 to VCC+0.3	V
Operating Temperature	TA	-20 to 70	°C
Storage Temperature	TSTG	-40 to 125	°C
Soldering Temperature/Time	TSOLDER	260°C/10 Sec	°C/Sec

9. Electrical Characteristics

DC Characteristics

(TA=25°C, unless otherwise noted)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
VCC	Operation Power Voltage		4.35	5.0	5.5	V
ICC(VCC)	Input VCC current on charge mode (regular operation)	VCC > VCC (min)		1	3	mA
ICC(Sleep)	Input VCC current on sleep mode	VBAT≧VCC; VCC is OFF			25	μA
VIH	Digital I/O input high voltage	VCC Voltage applied 4.35v to 5.5v	2.5		5.5	V
VIL	Digital I/O input low voltage	VCC Voltage applied 4.35v to 5.5v	-0.3		1.0	V
VOH	Digital I/O output high voltage	Reference to VCC	0.5		1.0	VCC
VOL	Digital I/O output low voltage	Reference to VCC	0.4		0.8	VCC
ISINK	Digital I/O output sink current	Output sink current of digital I/O pins set as output mode	10		50	mA
ISOURCE	Digital I/O output source current	Output source current of digital I/O pins set as input mode	0.1		1	mA
VREF	Internal reference voltage	The voltage select register VOSEL[3:0] = 4'b1100 (the register CVCTL at the data memory address=0BH), measured from voltage supply VCC region 4.35V to 5.50V	3.9		4.3	V
VCREF	Build in reference voltage temperature coefficient	TA=0~60°C		150		ppm/°C
FRC	Internal RC oscillator	External R=200kΩ		4		MHz

10. Functional Description

10.1 Typical Charging Scheme

10.1.1 Typical Charging Conditions and Phases

The FS3861 uses flexible control schemes of charger's current and voltage regulations in conjunction with the built-in 8-bit RISC-type MCU core running at typical 4 MHz for desired charge sequence controls during its operations. It is embedded with the constant-current and constant-voltage regulations as well as the additional facilities of PWM voltages for user-defined intermediate voltage levels used for various applications.

The external sensing resistors together with built-in parameters of the 8-bit MCU enable the device performing charge cycle operations through selections of small to larger charge current's amounts primarily for Li+ battery's linear mode charge applications, where the pulse-mode charging can be implemented using the internal hardware to control the charge sequences as implemented by the built-in MCU program code for various charger applications.

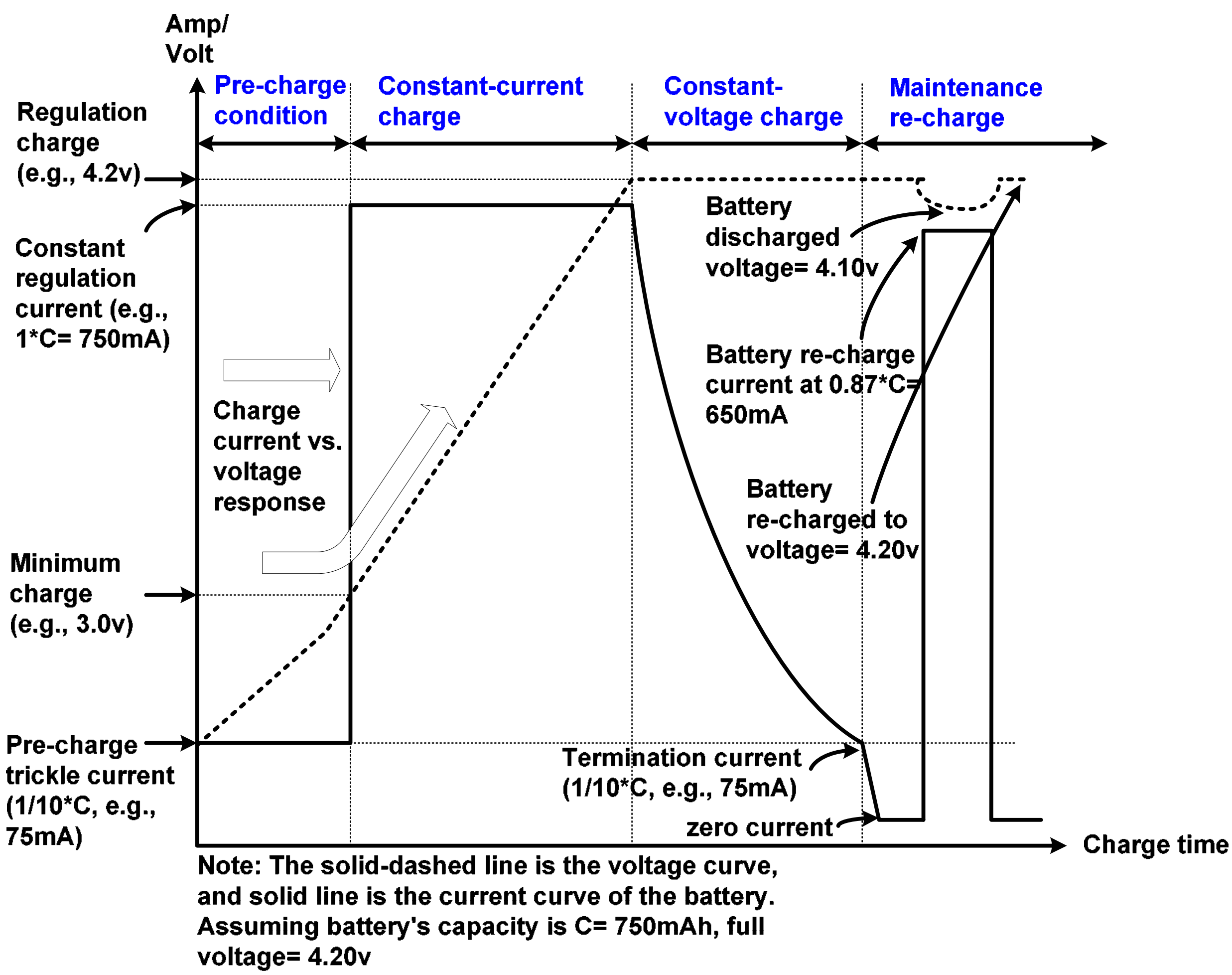


Fig.1 Typical Charge Profile

The typical Li+ charge steps are mainly four stages to conduct:

- **Pre-charge conditioning** (or called **trickle-mode charge,** as the Phase-0 stage): where the low-voltage discharged battery typical lower than 3.0v (or 2.8v, depending on how the battery's parameters are set) gets wake-up by applying typical 1/10 of full-rate charge current (a small amount of selectable charge current, also called trickle current, such as 85mA of 850mA charge current) until reaching the threshold voltage 3.0v. If the trickle current has been applied to the battery for more than 30 minutes by timer's measurement and not reaching the required 3.0v, it could be detected as bad battery without continuing to the next step of charge operations.
- **Constant current charge** (as Phase-1, referred as **C-C stage**): where the programmable constant current ranging from typical 250mA to 1,050mA is applied to the battery, until the battery voltage reaches to the full-level at 4.2v or similar value such as 4.1v or even 4.0v. Some applications require the constant current charge at USB current of 500mA when its power line at 5v is applied, and such charge stage can be implemented with selection of the current regulation at 500mA by setting the corresponding C-C reference bit and current select values at the specified control registers, as explained in details descriptions in later section.
- **Constant voltage charge** (shown as Phase-2, referred as **C-V stage**): using the regulated voltage at 4.2v reached at the constant current charge stage until the termination condition is met at the final low termination top-off current at smaller amount (such as 100mA which can be programmable to select), and then charges to the full capacity when termination occurs. Selections of the C-V charge's voltage level can be made with corresponding C-V enable and voltage select values at the individual specified control registers.
- **Maintenance re-charge** (shown as Phase-3 stage): can be called Post-charge stage, which is to resume charges to the battery when the battery's voltage drops is more than 0.1v (i.e. The battery terminal voltage becomes 4.10v or less from its full voltage at 4.20v) as a result of the internal resistor during its idle state through some time. If the battery has been taken off for use on its portable device, there is no re-charge check to conduct since the state transitions to the initial state without the battery itself.

  In some other cases, the preliminary charge stage which can be conducted as one step prior to the phase-0 to assure the battery to be through the charge sequences has working functions to perform. This stage would involve in applying constant-voltage charge pulses at defined level of 4.0v or so to the battery, which was examined to determine if it's at low voltage of 2.5v or less. The charge pulses applied to the battery for a short period of 15 intervals with 10 seconds high (at 4.0v voltage beats) and 5 seconds low (ground) each to examine if the battery voltage still remain low at 2.5v or less, which is then considered as defective and should be discarded.

Sometimes another additional check-up procedure follows the termination of the C-V stage to assure the battery in proper waiting stage for operation. That is to have the battery stay idle from its charge termination at full voltage of 4.20v (or 4.1v, depending on the battery's manufacturer's parameters). Then the battery stays in for additional 10 (or 15, also an adjustable parameter) minutes, and then its voltage is examined to assure the terminal voltage won't be decreased to lower than 4.05v (or 3.95v if the situation prevails), then the battery is also determined as a defective one without reliable performance since it could be losing more than 0.15v within a short period of just 10 (or 15) minutes. These check-up procedures are optional.

In brief summary, the typical Li+ battery charger's procedures could be summarized in the following few steps: pre-charge conditioning, constant-current (C-C), constant voltage (C-V) stage, charge termination and monitor to re-charge, etc. There might have some individual charge's current- or voltage-control schemes within the designated step to perform.

10.1.2 Charging Application Circuit

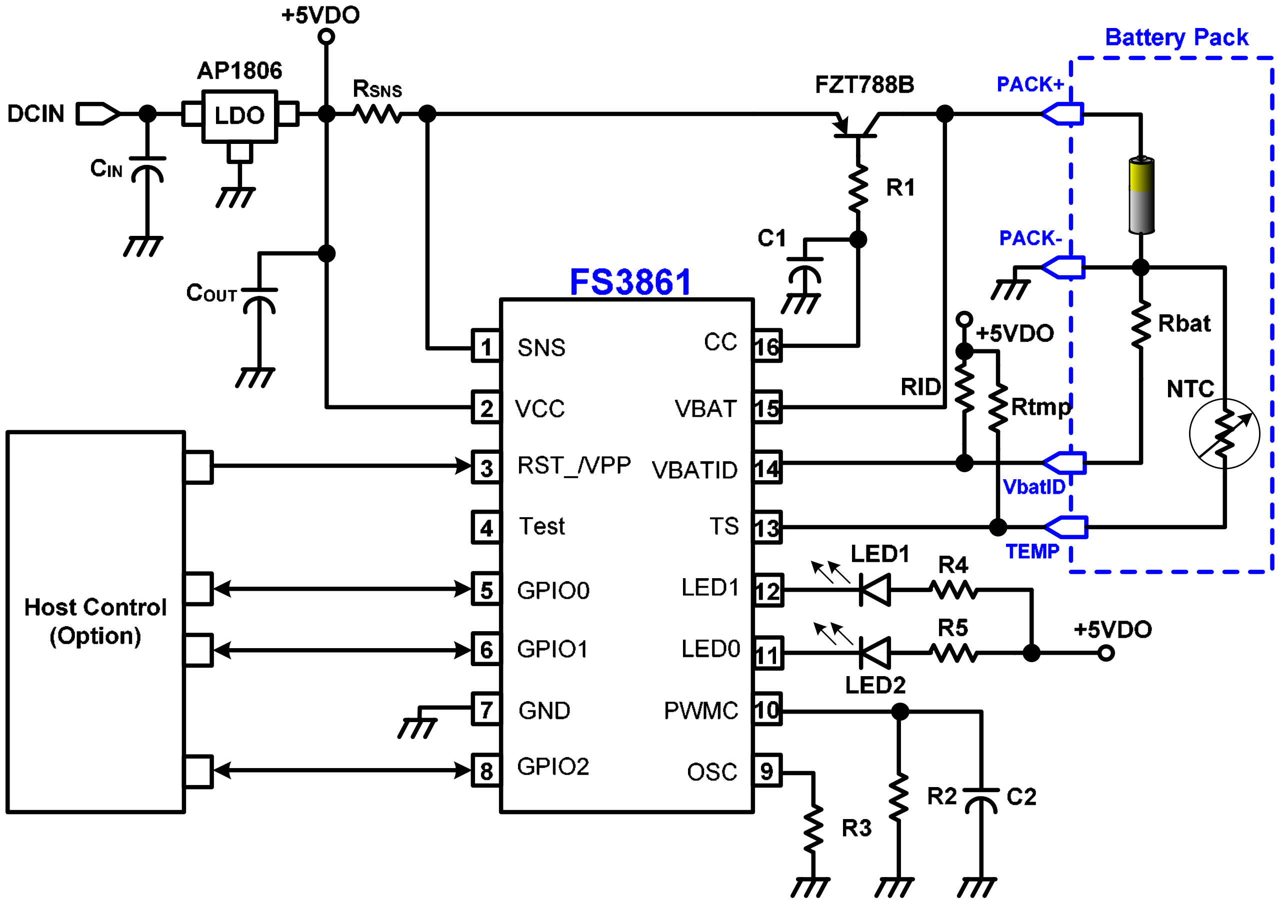


Fig.2 FS3861 Application Circuit

The Fig.2 shows the typical FS3861 application circuit used at base or travel charger devices of a variety of cellular phone and other portable devices. The above application circuitry shows the chip connected with an one-cell Li+ 4.2v battery, which features battery ID (at the VBATID input pin) and temperature sense output (at TS pin) for relevant controls. Interface to external host is optional at the general I/O bus pins with connections to the host side which commands the base charger with monitor facilities to control the charger operations. The use of PNP or PMOS as the pass transistor realizes the control of C-C and/or C-V mode current/voltage regulations.

10.1.3 Operation Flow Chart of Charging Application

Fig.3 is a typical example of operation state diagram.

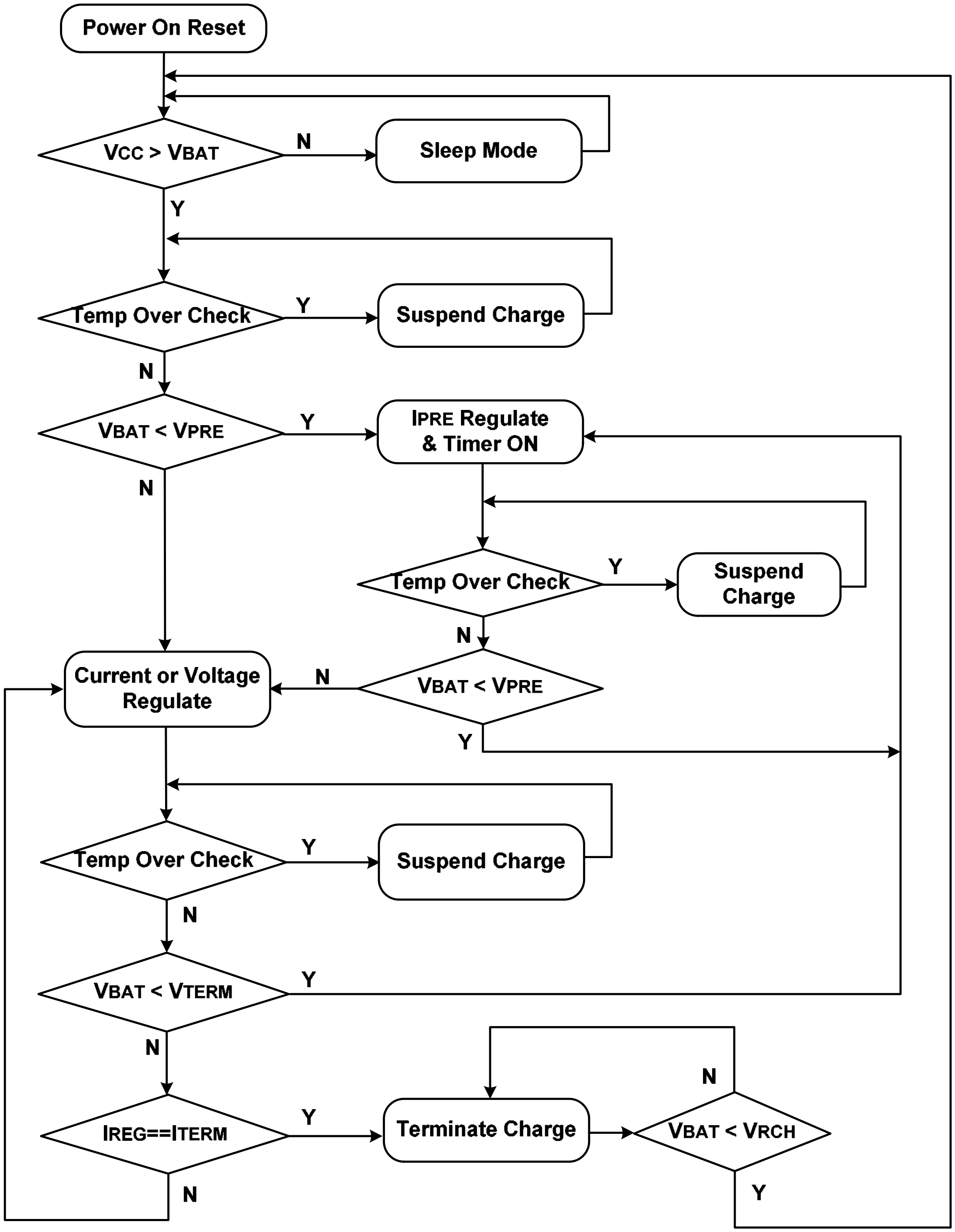


Fig.10-3 Typical operation flow chart

10.2 The Architecture of FS3861

The detailed architecture diagram of the FS3861 has already shown on Fig.7-1 for illustrations of its operations by the functional blocks, where the major facilities are constant-voltage (C-V) and constant-current (C-C) reference look-up table and regulation units as controlled by the MCU to realize the Li+ battery charge schemes. The FS3861 charger controller functions with illustrations of the current and voltage regulations, MCU, OTP ROM, and comparator implementing the linear-mode charge control.

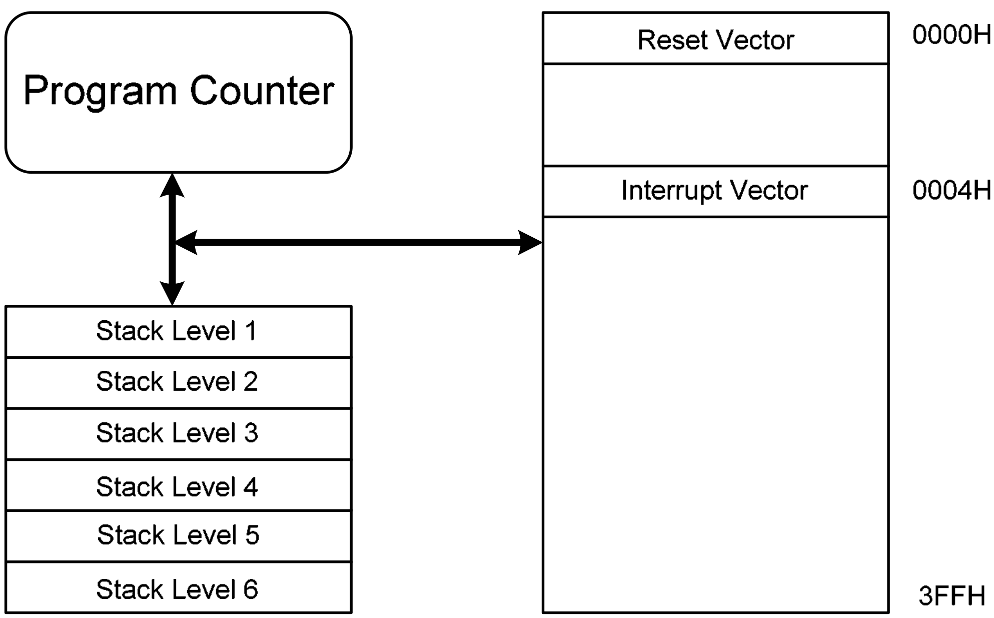
Note the built-in PWM (or called PDM, as named by the pulse density modulations) unit is complementary to the fixed voltage reference for proper generation of reference voltage to use in the intermediate charge control. Their VPWM levels subject to the PWM's setting of the fraction's bit and clock timing selects, as described in the later section of data memory register definitions, so the PWM's voltage level can then be used to perform specific voltage regulation in constant-voltage charge control to activate the output pin CC (charge control).

10.3 The organization of FS3861 MCU and its program & data memory space

The FS3861 charger controller employs FSC's proprietary RISC-architecture pipelined-mode high-performance 8-bit MCU core with built-in 1K Word program memory space and 64 Bytes of data memory space.

10.3.1 Program Memory Organization

CPU has a 10-bit program counter capable of address up to 1k x 16 program memory space. The reset vector is at 0000H and the interrupt vector is at 0004H.



10.3.2 Data Memory Organization

The data memory is partitioned into three parts. The address 00H~07H areas are system special registers, like indirect address, indirect address pointer, status register, working register, interrupt flag and interrupt control register. The address 08H~1FH areas are peripheral special registers. The address 80H~BFH areas are general data memory.

Address	Name	Content (u means unknown or unchanged)								Reset State	WDT Reset State
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0		
00H	IND0	Use contents of FSR0 to address data memory								uuuuuuuu	uuuuuuuu
01H	IND1	Use contents of FSR1 to address data memory								uuuuuuuu	uuuuuuuu
02H	FSR0	Indirect data memory, address pointer 0								uuuuuuuu	uuuuuuuu
03H	FSR1	Indirect data memory, address pointer 1								uuuuuuuu	uuuuuuuu
04H	STATUS	-	-	-	PD	-	DC	C	Z	uuu**0**uuuu	uuuuuuuu
05H	WORK	WORK register								uuuuuuuu	uuuuuuuu
06H	INTF	-	-	-	NORMIF	OVLOIF	UVLOIF	VDDIF	TMIF	uuu00000	uuu00000
07H	INTE	GIE	-	-	NORMIE	OVLOIE	UVLOIE	VDDIE	TMIE	0uu00000	0uu00000
08H~1FH		Peripheral special registers									
80H~BFH		General data Memory (64 bytes SRAM)									

10.3.2.1. System Special Registers

IND0, IND1 : ADDRESS 00H, 01H

The IND[1:0] registers at data memory address are not physical registers. Any instruction using the IND[1:0] registers actually access the data pointed by the FSR[1:0] registers.

bit7~0 Use contents of FSR0 (IND0: Address 00H) or FSR1 (IND1: Address 01H) to address data memory

FSR0, FSR1 : ADDRESS 02H, 03H

Indirect addressing pointers FSR0 and FSR1 correspond to IND0 and IND1 respectively.

bit7~0 Indirect data memory, address pointer 0 (Address 02H)
bit7~0 Indirect data memory, address pointer 1 (Address 03H)

STATUS : ADDRESS 04H

The STATUS register contains the arithmetic status of the ALU.

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
-	-	-	PD	-	DC	C	Z

bit 7~5 unimplemented
bit 4 **PD**: Power Down Flag
1 = After power on reset or cleared by writing 0 (which shuts off oscillator clock, thus neither of the MCU clock or operation will be in conduct)
0 = By execution of the SLEEP instruction, but not the HALT instruction (which only turns off the MCU clock)
bit 3 unimplemented

bit 2 **DC**: Digit Carry Flag (ADDWF, SUBWF instructions)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result
bit 1 **C**: Carry Flag (~Borrow)
bit 0 **Z**: Zero Flag
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

WORK : ADDRESS 05H
bit7~0 Store temporary data

INTF, INTE: ADDRESS 06H, 07H

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
-	-	-	NORMIF	OVLOIF	UVLOIF	VDDIF	TMIF
GIE	-	-	NORMIE	OVLOIE	UVLOIE	VDDIE	TMIE

bit7 **GIE**: Global interrupt enable (Address 07H)
bit6~5 unimplemented
bit4 **NORMIF**, **NORMIE**: VDD within normal working range (4.35V~5.5V) Interrupt flag and enable. NORMIF can wake up MCU if MCU is in sleep mode.
bit3 **OVLOIF**, **OVLOIE**: VDD over normal working range (VDD>5.5V) Interrupt flag and enable.
bit2 **UVLOIF**, **UVLOIE**: VDD under normal working range (VDD<4.35V) Interrupt flag and enable.
bit1 **VDDIF**, **VDDIE**: VDD > VBAT Interrupt flag and enable. Used when there is only VBAT and VDD is off. VDDIF can wake up MCU if MCU is in sleep mode.
bit0 **TMIF**, **TMIE**: 16-bit Timer Interrupt flag and enable.

10.3.2.2. Peripheral Special Registers

Address	Name	Content (u means unknown or unchanged)								Reset State	WDT Reset State
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0		
08H	POWER	ENBGR_	ENOVLO	-	-	ENCMP	-	-	-	00uu0uuu	00uu0uuu
09H	RESULT	-	OVLO	UVLO	NORM	CMPOUT	-	-	VDDIN	uuuuuuuu	uuuuuuuu
0AH	CCCTL	ENCCref	ENCC	-	-	CURSEL[3]	CURSEL[2]	CURSEL[1]	CURSEL[0]	00uu0000	00uu0000
0BH	CVCTL	ENCVref	ENCV	-	-	VOLSEL[3]	VOLSEL[2]	VOLSEL[1]	VOLSEL[0]	00uu0000	00uu0000
0CH	CMPSEL	CMPNSEL[2]	CMPNSEL[1]	CMPNSEL[0]	CMPPSEL[4]	CMPPSEL[3]	CMPPSEL[2]	CMPPSEL[1]	CMPPSEL[0]	00000000	00000000
0DH	PWDH	PWM[15]	PWM [14]	PWM [13]	PWM [12]	PWM [11]	PWM [10]	PWM [9]	PWM [8]	00000000	00000000
0EH	PWDL	PWM[7]	PWM[6]	PWM [5]	PWM [4]	PWM [3]	PWM [2]	PWM [1]	PWM [0]	00000000	00000000
0FH	PWDCON	-	-	-	PWEN	-	PWCS[2]	PWCS[1]	PWCS[0]	uuu0u000	uuu0u000
10H	TMOUT	TMOUT[7:0]								00000000	00000000
11H	TMCON	TRST	-	-	-	TMEN	INS[2]	INS[1]	INS[0]	uuuu0000	uuuu0000
12H	LEDCTL	LED1EN	LED1	LED0EN	LED0	-	GPIO2OEN	GPIO2	GPIO2PU	0000u000	0000u000*
13H	GPIO	SPWMEN	GPIO1OEN	GPIO1	GPIO1PU	SPWMO	GPIO0OEN	GPIO0	GPIO0PU	00000000	00000000*
14H	-	Unimplemented								uuuuuuuu	
15H	-	Unimplemented								0uuuu000	0uuuu000

***** Input Mode doesn't pull up.

POWER : ADDRESS 08H

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
ENBGR_	ENOVLO	-	-	ENCMP	-	-	-

bit7 **ENBGR_**: Enable the internal bandgap references of both the voltage and current regulations. This bit is active LOW enable. Thus, the procedure of enabling the current or voltage regulations is to enable the **ENBGR_** before enabling the **ENCCref**, **ENCC (ADDRESS 0AH)** and **ENCVref**, **ENCV (ADDRESS 0BH)**.

bit6 **ENOVLO**: Enable the working voltage detection to assure the input voltage satisfied 4.35V < VCC < 5.5V.

bit5~4 unimplemented

bit3 **ENCMP** : The comparator enable bit enables the internal comparator for comparison between the measured input parameters (such as the battery voltage, sensed charged current, battery temperature, etc.) and the pre-set current or voltage values selected by the comparator select **CMPPSEL[4:0] (ADDRESS 0CH)**.

bit2~0 unimplemented

RESULT : ADDRESS 09H

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
-	OVLO	UVLO	NORM	CMPOUT	-	-	VDDIN

bit7 unimplemented

bit6 **OVLO**: Over voltage lock-out status READ ONLY register bit.

bit5 **UVLO**: Under voltage lock-out status READ ONLY register bit.

bit4 **NORM**: Normal status READ ONLY register bit.

bit3 **CMPOUT**: The comparator output.
bit2~1 unimplemented
bit0 **VDDIN**: The status indicator (Read Only) of supply voltage VCC greater than VBAT, i.e. VDDIN is set high when VCC > VBAT. If there is no VCC and only the VBAT of battery is connected, then the VDDIN is set inactive low.

CCCTL : ADDRESS 0AH

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
ENCCref	ENCC	-	-	CURSEL[3]	CURSEL[2]	CURSEL[1]	CURSEL[0]

bit7 **ENCCref**: Enable constant current regulation reference current.
bit6 **ENCC**: Enables the constant current regulation for constant current charge with desired current amount selected.
bit5~4 unimplemented
bit[3:0] The 3-bits select **CURSEL[3:0]** selects constant current regulation reference current. The regulation current accuracy is 10% (unless otherwise noted).

CURSEL[3:0]	0000	0001	0010	0011	0100	0101	0110	0111
Select	**45mA ±20mA**	**70mA ±20mA**	**95mA ±20mA**	**120mA ±20mA**	**145mA ±20mA**	**170mA ±20mA**	**265mA**	**360mA**
CURSEL[3:0]	1000	1001	1010	1011	1100	1101	1110	1111
Select	**410mA**	**460mA**	**500mA**	**650mA**	**750mA**	**850mA**	**950mA**	**1050mA**

CVCTL : ADDRESS 0BH

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
ENCVref	ENCV	-	-	VOLSEL[3]	VOLSEL[2]	VOLSEL[1]	VOLSEL[0]

bit7 **ENCVref**: Enable constant voltage regulation reference current.
bit6 **ENCV**: Enables the constant voltage regulation for constant voltage charge with desired voltage amount selected.
bit5~4 unimplemented
bit[3:0] The 3-bits **VOLSEL[3:0]** selects constant voltage regulation reference voltage. The regulation voltage accuracy is ±1%.

VOLSEL [3:0]	0000	0001	0010	0011	0100	0101	0110	0111
Select	**1.2V**	**1.4V**	**2.4V**	**2.8V**	**3.2V**	**3.4V**	**3.6V**	**4.0V**
VOLSEL [3:0]	1000	1001	1010	1011	1100	1101	1110	1111
Select	**4.1V**	**4.2V**	**4.23V**	**4.25V**	**4.3V**	**VPWM**	Reserved	Reserved

CMPSEL : ADDRESS 0CH

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
CMPNSEL[2]	CMPNSEL[1]	CMPNSEL[0]	CMPPSEL[4]	CMPPSEL[3]	CMPPSEL[2]	CMPPSEL[1]	CMPPSEL[0]

Also refer to **ENCMP (ADDRESS 08H)** and **CMPOUT (ADDRESS 09H)** register bits. The bit CMPOUT is the compared output and would be set high on comparator's measured input value (like current across the sense resistor Rsns with selecting the desired measured item by setting the **CMPNSEL[2:0]**) match exactly with the positive input of the selected reference value.

bit[7:5] **CMPNSEL[2:0]** select comparator negative input.

CMPNSEL [2:0]	000	001	010	011	100	101	110	111
Select	**TS**	**VBATID**	**VBAT**	**SNS**	**ICTEMP**	Reserved	Reserved	Reserved

TS The voltage of external thermistor, with either PTC (positive temperature) or NTC (negative temperature) coefficient, and is compared with VPWM for temperature measurements and subsequent control actions.

VBATID The voltage of external Battery ID, and is compared with VPWM for determining the battery's types before charges.

VBAT Battery voltage and will be compared with 2.5V, 2.6V, 3.0V, 3.9V,….4.25V, 4.3V, also shown on the voltage regulations.

SNS The current sensing voltage and will be compared with the 50mA, 100mA, 150mA to determine the termination current.

ICTEMP The chips die body temperature measurement for prevention of excessive heat while in continuous operations.

bit[4:0] **CMPPSEL[4:0]** select comparator positive input.

CMPPSEL [4:0]	00000	00001	00010	00011	00100	00101	00110	00111
Select	**VPWM**	**1.2V**	**1.4V**	**1.5V**	**2.0V**	**2.4V**	**2.5V**	**2.6V**
CMPPSEL [4:0]	01000	01001	01010	01011	01100	01101	01110	01111
Select	**2.8V**	**3.0V**	**3.1V**	**3.2V**	**3.3V**	**3.4V**	**3.6V**	**3.9V**
CMPPSEL [4:0]	10000	10001	10010	10011	10100	10101	10110	10111
Select	**4.0V**	**4.10V**	**4.15V**	**4.20V**	**4.23V**	**4.25V**	**4.3V**	**60mA ±20mA**
CMPPSEL [4:0]	11000	11001	11010	11011	11100	11101	11110	11111
Select	**85mA ±20mA**	**110mA ±20mA**	**130mA ±20mA**	**160mA ±20mA**	**185mA ±20mA**	**265mA ±10%**	**360mA ±10%**	**410mA ±10%**

10.3.2.3. PWM (PDM) Voltage Generation

The VPWM which is one of the comparator’s selected item by setting the **CMPPSEL[4:0]**==5’b00000 is the voltage level generated by the PWM function(pulse-width-modulation, or called pulse-density-modulation abbreviated as PDM, since the scheme here is using the PDM instead of the PWM for generating pulse clock signals), either selected by the hardware or software PWM module. The pulse density modulation clock enable bits used at hardware PWM mode can be selected on the contents of the registers PWDH[7:0] and PWDL[7:0] addressed at 0DH and 0EH, respectively.

PWDH : ADDRESS 0DH

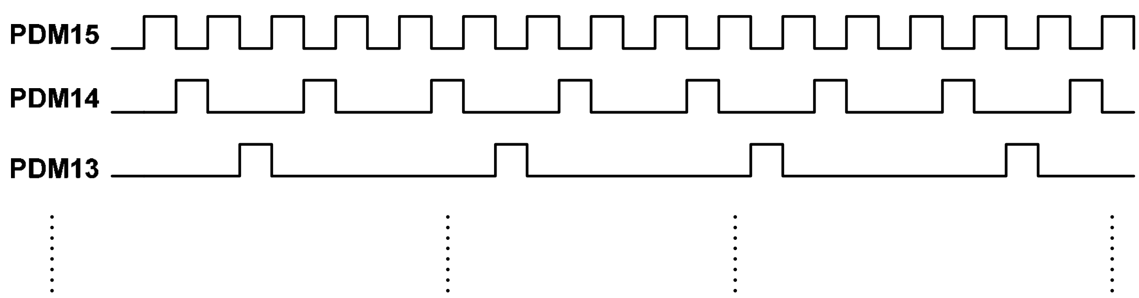
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
PWM[15]	PWM [14]	PWM [13]	PWM [12]	PWM [11]	PWM [10]	PWM [9]	PWM [8]

PWDL : ADDRESS 0EH

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
PWM[7]	PWM [6]	PWM [5]	PWM [4]	PWM [3]	PWM [2]	PWM [1]	PWM [0]

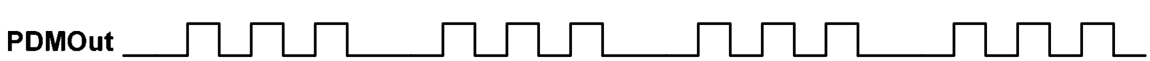
PWM[15:0]={PWDH[7:0],PWDL[7:0]}

The PWM reference voltage is generated by using the derived divider chained-clocks for generation of the sub-digit voltage level. For example, the bit PWM [14] = PWDH [6] =1’b1 refers to the clock-chained derived clock to make the voltage-level divide-by-4, i.e., $V_{CC} / 2^{-2} = V_{CC} / 4$. The following figure shows the PDM definition:



PDMOut = PDM[15] OR PDM[14] OR ..... OR PDM[0]

Ex: PDMOut = 6000h



From above, we know that PDM[15] represents the same energy weighting of PWM[15] in the 16-bit period of time. PDM[15] can generate the same 32,768 counts of positive pulse, (PDM[15] = 1 = PWM[15]) or 0 count (PDM[15] = 0 = PWM[15]) as normal PWM[15] does in the 16-bit period of time. Also, PDM[14] can generate the same 16,384 counts of positive pulse, (PDM[14] = 1 = PWM[14]) or 0 count (PDM[14] = 0 = PWM[14]) as PWM[14] does, and so on. Then, we know that we may get the same energy weighting (or counts of positive pulse) in the 16-bit period of time if we set the same value on PDM[15:0] and PWM[15:0]. If we zoom into the 8-bit period of time from the beginning within the 16-bit period with the setting of PDM[15:0]= 1000-0000-0000-0000b = PWM[15:0], we will see that PDM offers better energy transformation than PWM does. PDM still offers half energy (128 counts of positive pulse) within the 8-bit period of time from the beginning within the 16-bit period, but PWM offers full energy (256 counts of positive pulse) within the same period.

For example, if the PWM [15:0] =30A4H, then the voltage level is Full-Scale*(30A4H / FFFFH) = 0.19*Full-Scale and vice versa. Also note that the software PWM enable bit **SPMWEN (ADDRESS 13H)** should be set inactive low to enable the hardware PWM for the PWM generated specific voltage.

PWDCON : ADDRESS 0FH

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
-	-	-	PWEN	-	PWCS[2]	PWCS[1]	PWCS[0]

bit[7:5] unimplemented
bit4 **PWEN**: Enable Pulse Density Modulation clock output.
bit3 unimplemented
bit[2:0] **PWCS[2:0]** selects Pulse Density Modulation clock input source. Setting as below:

PWCS [2:0]	000	001	010	011	100	101	110	111
Select	-	-	-	-	4MHz/32	4MHz/64	4MHz/128	4MHz/256

10.3.2.4. Timer Interrupt Register, LED output displays, and General I/O data bits

TMOUT : ADDRESS 10H

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
TMOUT [7:0]							

TMOUT [7:0] is the output of the 8-bit counter, read-only register.

TMCON : ADDRESS 11H

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
TRST	-	-	-	TMEN	INS[2]	INS[1]	INS[0]

bit7 **TRST**: If set TRST=0, the MCU will reset the 8-bit counter. Then read TRST bit will get "1".
bit[6:4] unimplemented
bit3 **TMEN**: Counter enable
1 = The 8-bit counter will be enabled
0 = The 8-bit counter will be disabled
bit2[2:0] **INS[2:0]** selects timer interrupt source **TMOUT[7:0]** while Timer Clock source = 4MHz/32 = 128 kHz.

INS[2:0] selects the interrupt source TMOUT[7:0], as shown below, where the timer clock's master source is kept at 128KHz corresponding to INS[2:0]==3'b111. For example, if we want to use the timer clock at 64kHz, then INS[2:0]== 3'b110 should be set to get the corresponding timer clock; and if we want the timer activated at the divide-by-32 corresponding to TMOUT[7:0]==8'h20, the interrupt would be activated whenever the 64kHz clock has the count equal to 32 (or 20H).

INS [2:0]	000	001	010	011	100	101	110	111
Interrupt Source	TMOUT[0]	TMOUT[1]	TMOUT[2]	TMOUT[3]	TMOUT[4]	TMOUT[5]	TMOUT[6]	TMOUT[7]

TMOUT[7:0]	TMOUT[0]	TMOUT[1]	TMOUT[2]	TMOUT[3]	TMOUT[4]	TMOUT[5]	TMOUT[6]	TMOUT[7]
Interrupt	Timer Clock Source	Timer Clock Source/2	Timer Clock Source/4	Timer Clock Source/8	Timer Clock Source/16	Timer Clock Source/32	Timer Clock Source/64	Timer Clock Source/128

LEDCTL : ADDRESS 12H

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
LED1EN	LED1	LED0EN	LED0	-	GPIO2OEN	GPIO2	GPIO2PU

bit7 **LED1EN**: LED1 enable bit
1 = Enabled LED1 control
0 = Disable LED1 control
bit6 **LED1**: LED1 output control
1 = Enable LED1 sink output (10mA)
0 = not used
bit5 **LED0EN**: LED0 enable bit
1 = Enabled LED0 control
0 = Disable LED0 control

bit4 **LED0**: LED0 output control
1 = Enable LED0 sink output (10mA)
0 = not used
bit3 unimplemented
bit2 **GPIO2OEN**: GPIO2 output enable bit
1 = Enabled GPIO2 output
0 = Disable GPIO2 output
bit1 **GPIO2**: GPIO2 output H/L
bit0 **GPIO2PU**: Internal pull up 10kΩ.

GPIO : ADDRESS 13H

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
SPWMEN	GPIO1OEN	GPIO1	GPIO1PU	SPWMO	GPIO0OEN	GPIO0	GPIO0PU

bit7 **SPMWEN**: PWM control
1 = Enabled Software PWM control
0 = Enable Hardware PWM
bit6 **GPIO1OEN**: GPIO1 output enable bit
1 = Enabled GPIO1 output
0 = Disable GPIO1 output
bit5 **GPIO1**: GPIO1 output H/L
bit4 **GPIO1PU**: Internal pull up 10kΩ.
bit3 **SPWMO**: Software PWM H/L(“1” / ”0”) control bit.
bit2 **GPIO0OEN**: GPIO0 output enable bit
1 = Enabled GPIO0 output
0 = Disable GPIO0 output
bit1 **GPIO0**: GPIO0 output H/L
bit0 **GPIO0PU**: Internal pull up 10kΩ.

Not used : ADDRESS 14H

Not used : ADDRESS 15H

11. Instruction Set

The FS3861 instruction set consists of 37 instructions. Each instruction is a 16-bit word with an OPCODE and one or more operands. The detailed descriptions are shown as below.

11.1 Instruction Set Summary

Table11-1: FS3861 Instruction Set

Instruction	Operation	Cycle	Flag
ADDLW k	[W] ← [W] + k	1	C, DC, Z
ADDPCW	[PC] ← [PC] + 1 + [W]	2	None
ADDWF f, d	[Destination] ← [f] + [W]	1	C, DC, Z
ADDWFC f, d	[Destination] ← [f] + [W] + C	1	C, DC, Z
ANDLW k	[W] ← [W] AND k	1	Z
ANDWF f, d	[Destination] ← [W] AND [f]	1	Z
BCF f, b	[f<b>] ← 0	1	None
BSF f, b	[f<b>] ← 1	1	None
BTFSC f, b	Skip if [f<b>] = 0	1, 2	None
BTFSS f, b	Skip if [f<b>] = 1	1, 2	None
CALL k	Push PC + 1 and GOTO k	2	None
CLRF f	[f] ← 0	1	Z
CLRWDT	Clear watch dog timer	1	None
COMF f, d	[f] ← NOT([f])	1	Z
DECF f, d	[Destination] ← [f] -1	1	Z
DECFSZ f, d	[Destination] ← [f] -1, skip if the result is zero	1, 2	None
GOTO k	PC ← k	2	None
HALT	CPU Stop	1	None
INCF f, d	[Destination] ← [f] +1	1	Z
INCFSZ f, d	[Destination] ← [f] + 1, skip if the result is zero	1, 2	None
IORLW k	[W] ← [W] | k	1	Z
IORWF f, d	[Destination] ← [W] | [f]	1	Z
MOVFW f	[W] ← [f]	1	None
MOVLW k	[W] ← k	1	None
MOVWF f	[f] ← [W]	1	None
NOP	No operation	1	None
RETFIE	Pop PC and GIE = 1	2	None
RETLW k	RETURN and W = k	2	None
RETURN	Pop PC	2	None
RLF f, d	[Destination<n+1>] ← [f<n>]	1	C,Z
RRF f, d	[Destination<n-1>] ← [f<n>]	1	C, Z
SLEEP	Stop OSC	1	PD
SUBLW k	[W] ← k – [W]	1	C, DC, Z
SUBWF f, d	[Destination] ← [f] – [W]	1	C, DC, Z
SUBWFC f, d	[Destination] ← [f] – [W] – Ċ	1	C, DC, Z
XORLW k	[W] ← [W] XOR k	1	Z
XORWF f, d	[Destination] ← [W] XOR [f]	1	Z

Note:

- f: memory address (00h ~ 7Fh).
- W: work register.
- k: literal field, constant data or label.
- d: destination select: d=0 store result in W, d=1: store result in memory address f.
- b: bit select (0~7).
- [f]: the content of memory address f.
- PC: program counter.
- C: Carry flag
- DC: Digit carry flag
- Z: Zero flag
- PD: power down flag
- TO: watchdog time out flag
- WDT: watchdog timer counter

11.2 Instruction Description

(By alphabetically)

ADDLW	Add Literal to W
Syntax	ADDLW k $0 \le k \le FFh$
Operation	[W] ← [W] + k
Flag Affected	C, DC, Z
Description	The content of Work register add literal “k” in Work register
Cycle	1
Example: **ADDLW 08h**	Before instruction: W = 08h After instruction: W = 10h

ADDPCW	Add W to PC
Syntax	ADDPCW
Operation	[PC] ← [PC] + 1 + [W], [W] < 79h [PC] ← [PC] + 1 + ([W] – 100h), otherwise
Flag Affected	None
Description	The relative address PC + 1 + W are loaded into PC.
Cycle	2
Example 1: **ADDPCW**	Before instruction: W = 7Fh, PC = 0212h After instruction: PC = 0292h
Example 2: **ADDPCW**	Before instruction: W = 80h, PC = 0212h After instruction: PC = 0193h
Example 3: **ADDPCW**	Before instruction: W = FEh, PC = 0212h After instruction: PC = 0211h

ADDWF	Add W to f
Syntax	ADDWF f, d 0 ≤ f ≤ FFh d ∈ [0,1]
Operation	[Destination] ← [f] + [W]
Flag Affected	C, CD, Z
Description	Add the content of the W register and [f]. If d is 0, the result is stored in the W register. If d is 1, the result is stored back in f.
Cycle	1
Example 1: **ADDWF OPERAND, 0**	Before instruction: OPERAND = C2h W = 17h After instruction: OPERAND = C2h W = D9h
Example 2: **ADDWF OPERAND, 1**	Before instruction: OPERAND = C2h W = 17h After instruction: OPERAND = D9h W = 17h

ADDWFC	Add W, f and Carry
Syntax	ADDWFC f, d 0 ≤ f ≤ FFh d ∈ [0,1]
Operation	[Destination] ← [f] + [W] + C
Flag Affected	C, DC, Z
Description	Add the content of the W register, [f] and Carry bit. If d is 0, the result is stored in the W register. If d is 1, the result is stored back in f.
Cycle	1
Example **ADDWFC OPERAND,1**	Before instruction: C = 1 OPERAND = 02h W = 4Dh After instruction: C = 0 OPERAND = 50h W = 4Dh

ANDLW	AND literal with W
Syntax	ANDLW k 0 ≤ k ≤ FFh
Operation	[W] ← [W] AND k
Flag Affected	Z
Description	AND the content of the W register with the eight-bit literal "k". The result is stored in the W register.
Cycle	1
Example: **ANDLW 5Fh**	Before instruction: W = A3h After instruction: W = 03h

ANDWF	AND W and f
Syntax	ANDWF f, d 0 ≤ f ≤ FFh d ∈ [0,1]
Operation	[Destination] ← [W] AND [f]
Flag Affected	Z
Description	AND the content of the W register with [f]. If d is 0, the result is stored in the W register. If d is 1, the result is stored back in f.
Cycle	1
Example 1: **ANDWF OPERAND,0**	Before instruction: W = 0Fh, OPERAND = 88h After instruction: W = 08h, OPERAND = 88h
Example 2: **ANDWF OPERAND,1**	Before instruction: W = 0Fh, OPERAND = 88h After instruction: W = 88h, OPERAND = 08h

BCF	Bit Clear f
Syntax	BCF f, b 0 ≤ f ≤ FFh 0 ≤ b ≤ 7
Operation	[f<b>] ← 0
Flag Affected	None
Description	Bit b in [f] is reset to 0.
Cycle	1
Example: **BCF FLAG, 2**	Before instruction: FLAG = 8Dh After instruction: FLAG = 89h

BSF	Bit Set f
Syntax	BSF f, b 0 ≤ f ≤ FFh 0 ≤ b ≤ 7
Operation	[f<b>] ← 1
Flag Affected	None
Description	Bit b in [f] is set to 1.
Cycle	1
Example: **BSF FLAG, 2**	Before instruction: FLAG = 89h After instruction: FLAG = 8Dh

BTFSC	Bit Test skip if Clear
Syntax	BTFSC f, b 0 ≤ f ≤ FFh 0 ≤ b ≤ 7
Operation	Skip if [f<b>] = 0
Flag Affected	None
Description	If bit 'b' in [f] is 0, the next fetched instruction is discarded and a NOP is executed instead of making it a two-cycle instruction.
Cycle	1, 2
Example: Node **BTFSC FLAG, 2** OP1 : OP2 :	Before instruction: PC = address (Node) After instruction: If FLAG<2> = 0 PC = address(OP2) If FLAG<2> = 1 PC = address(OP1)

BTFSS	Bit Test skip if Set
Syntax	BTFSS f, b 0 ≤ f ≤ FFh 0 ≤ b ≤ 7
Operation	Skip if [f<b>] = 1
Flag Affected	None
Description	If bit 'b' in [f] is 1, the next fetched instruction is discarded and a NOP is executed instead of making it a two-cycle instruction.
Cycle	1, 2
Example: Node **BTFSS FLAG, 2** OP1 : OP2 :	Before instruction: PC = address (Node) After instruction: If FLAG<2> = 0 PC = address(OP1) If FLAG<2> = 1 PC = address(OP2)

CALL	Subroutine CALL
Syntax	CALL k 0 ≤ k ≤ 1FFFh
Operation	Push Stack [Top Stack] ← PC + 1 PC ← k
Flag Affected	None
Description	Subroutine Call. First, return address PC + 1 is pushed onto the stack. The immediate address is loaded into PC.
Cycle	2

CLRF	Clear f
Syntax	CLRF f 0 ≤ f ≤ 255
Operation	[f] ← 0
Flag Affected	None
Description	Reset the content of memory address f
Cycle	1
Example: **CLRF WORK**	Before instruction: WORK = 5Ah After instruction: WORK = 00h

CLRWDT	Clear watch dog timer
Syntax	CLRWDT
Operation	Watch dog timer counter will be reset
Flag Affected	None
Description	CLRWDT instruction will reset watch dog timer counter.
Cycle	1
Example: **CLRWDT**	After instruction: WDT = 0 TO = 1 PD = 1

COMF	Complement f
Syntax	COMF f, d $0 \leq f \leq 255$ $d \in [0,1]$
Operation	[f] ← NOT([f])
Flag Affected	Z
Description	[f] is complemented. If d is 0, the result is stored in the W register. If d is 1, the result is stored back in [f]
Cycle	1
Example 1: **COMF OPERAND,0**	Before instruction: W = 88h, OPERAND = 23h After instruction: W = DCh, OPERAND = 23h
Example 2: **COMF OPERAND,1**	Before instruction: W = 88h, OPERAND = 23h After instruction: W = 88h, OPERAND = DCh

DECF	Decrement f
Syntax	DECF f, d $0 \leq f \leq 255$ $d \in [0,1]$
Operation	[Destination] ← [f] -1
Flag Affected	Z
Description	[f] is decremented. If d is 0, the result is stored in the W register. If d is 1, the result is stored back in [f].
Cycle	1
Example 1: **DECF OPERAND,0**	Before instruction: W = 88h, OPERAND = 23h After instruction: W = 22h, OPERAND = 23h
Example 2: **DECF OPERAND,1**	Before instruction: W = 88h, OPERAND = 23h After instruction: W = 88h, OPERAND = 22h

DECFSZ	Decrement f, skip if zero
Syntax	DECFSZ f, d $0 \leq f \leq FFh$ $d \in [0,1]$
Operation	[Destination] ← [f] -1, skip if the result is zero
Flag Affected	None
Description	[f] is decremented. If d is 0, the result is stored in the W register. If d is 1, the result is stored back in [f]. If the result is 0, then the next fetched instruction is discarded and a NOP is executed instead of making it a two-cycle instruction.
Cycle	1, 2
Example: Node **DECFSZ FLAG, 1** OP1 : OP2 :	Before instruction: PC = address (Node) After instruction: [FLAG] = [FLAG] - 1 If [FLAG] = 0 PC = address(OP1) If [FLAG] ≠ 0 PC = address(OP2)

GOTO	Unconditional Branch
Syntax	GOTO k $0 \le k \le 1FFFh$
Operation	PC ← k
Flag Affected	None
Description	The immediate address is loaded into PC.
Cycle	2

HALT	Stop CPU Core Clock
Syntax	HALT
Operation	CPU Stop
Flag Affected	None
Description	CPU clock is stopped. Oscillator is running. CPU can be waked up by internal and external interrupt sources.
Cycle	1

INCF	Increment f
Syntax	INCF f, d $0 \le f \le FFh$ $d \in [0,1]$
Operation	[Destination] ← [f] +1
Flag Affected	Z
Description	[f] is incremented. If d is 0, the result is stored in the W register. If d is 1, the result is stored back in [f].
Cycle	1
Example 1: **INCF OPERAND,0**	Before instruction: W = 88h, OPERAND = 23h After instruction: W = 24h, OPERAND = 23h
Example 2: **INCF OPERAND,1**	Before instruction: W = 88h, OPERAND = 23h After instruction: W = 88h, OPERAND = 24h

INCFSZ	Increment f, skip if zero
Syntax	INCFSZ f, d $0 \le f \le FFh$ $d \in [0,1]$
Operation	[Destination] ← [f] + 1, skip if the result is zero
Flag Affected	None
Description	[f] is incremented. If d is 0, the result is stored in the W register. If d is 1, the result is stored back in [f]. If the result is 0, then the next fetched instruction is discarded and a NOP is executed instead of making it a two-cycle instruction.
Cycle	1, 2
Example: Node **INCFSZ FLAG, 1** OP1 : OP2 :	Before instruction: PC = address (Node) After instruction: [FLAG] = [FLAG] + 1 If [FLAG] = 0 PC = address(OP2) If [FLAG] ≠ 0 PC = address(OP1)

IORLW	Inclusive OR literal with W
Syntax	IORLW k $0 \le k \le FFh$
Operation	[W] ← [W] | k
Flag Affected	Z
Description	Inclusive OR the content of the W register and the eight-bit literal "k". The result is stored in the W register.
Cycle	1
Example: **IORLW 85H**	Before instruction: W = 69h After instruction: W = EDh

IORWF	Inclusive OR W with f
Syntax	IORWF f, d $0 \le f \le FFh$ $d \in [0,1]$
Operation	[Destination] ← [W] | [f]
Flag Affected	Z
Description	Inclusive OR the content of the W register and [f]. If d is 0, the result is stored in the W register. If d is 1, the result is stored back in [f].
Cycle	1
Example: **IORWF OPERAND,1**	Before instruction: W = 88h, OPERAND = 23h After instruction: W = 88h, OPERAND = ABh

MOVFW	Move f to W
Syntax	MOVFW f 0 ≤ f ≤ FFh
Operation	[W] ← [f]
Flag Affected	None
Description	Move data from [f] to the W register.
Cycle	1
Example: **MOVFW OPERAND**	Before instruction: W = 88h, OPERAND = 23h After instruction: W = 23h, OPERAND = 23h

MOVLW	Move literal to W
Syntax	MOVLW k 0 ≤ k ≤ FFh
Operation	[W] ← k
Flag Affected	None
Description	Move the eight-bit literal "k" to the content of the W register.
Cycle	1
Example: **MOVLW 23H**	Before instruction: W = 88h After instruction: W = 23h

MOVWF	Move W to f
Syntax	MOVWF f 0 ≤ f ≤ FFh
Operation	[f] ← [W]
Flag Affected	None
Description	Move data from the W register to [f].
Cycle	1
Example: **MOVWF OPERAND**	Before instruction: W = 88h, OPERAND = 23h After instruction: W = 88h, OPERAND = 88h

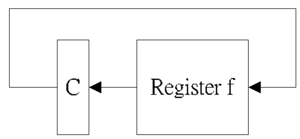
NOP	No Operation
Syntax	NOP
Operation	No Operation
Flag Affected	None
Description	No operation. NOP is used for one instruction cycle delay.
Cycle	1

RETFIE	Return from Interrupt
Syntax	RETFIE
Operation	[Top Stack] => PC Pop Stack 1 => GIE
Flag Affected	None
Description	The program counter is loaded from the top stack, then pop stack. Setting the GIE bit enables interrupts.
Cycle	2

RETLW	Return and move literal to W
Syntax	RETLW k 0 ≤ k ≤ FFh
Operation	[W] ← k [Top Stack] => PC Pop Stack
Flag Affected	None
Description	Move the eight-bit literal "k" to the content of the W register. The program counter is loaded from the top stack, then pop stack.

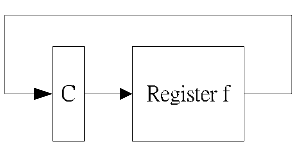
Return	Return from Subroutine
Syntax	RETURN
Operation	[Top Stack] => PC Pop Stack
Flag Affected	None
Description	The program counter is loaded from the top stack, then pop stack.
Cycle	2

RLF	Rotate left [f] through Carry
Syntax	RLF f, d 0 ≤ f ≤ FFh d ∈ [0,1]
Operation	[Destination<n+1>] ← [f<n>] [Destination<0>] ← C C ← [f<7>]
Flag Affected	C, Z
Description	[f] is rotated one bit to the left through the Carry bit. If d is 0, the result is stored in the W register. If d is 1, the result is stored back in [f].



Cycle	1
Example: **RLF OPERAND, 1**	Before instruction: C = 0 W = 88h, OPERAND = E6h After instruction: C = 1 W = 88h, OPERAND = CCh

RRF	Rotate right [f] through Carry
Syntax	RRF f, d 0 ≤ f ≤ FFh d ∈ [0,1]
Operation	[Destination<n-1>] ← [f<n>] [Destination<7>] ← C C ← [f<7>]
Flag Affected	C
Description	[f] is rotated one bit to the right through the Carry bit. If d is 0, the result is stored in the W register. If d is 1, the result is stored back in [f].



Cycle	1
Example: **RRF OPERAND, 0**	Before instruction: C = 0 OPERAND = 95h After instruction: C = 1 W = 4Ah, OPERAND = 95h

SLEEP	Oscillator stop
Syntax	SLEEP
Operation	CPU oscillator is stopped
Flag Affected	PD
Description	CPU oscillator is stopped. CPU can be waked up by external interrupt sources.
Cycle	1

- Please make sure that all interrupt flags are cleared before running SLEEP; "NOP" command must follow HALT and SLEEP commands.

SUBLW	Subtract W from literal
Syntax	SUBLW k 0 ≤ k ≤ FFh
Operation	[W] ← k – [W]
Flag Affected	C, DC, Z
Description	Subtract the content of the W register from the eight-bit literal "k". The result is stored in the W register.
Cycle	1
Example 1: **SUBLW 02H**	Before instruction: W = 01h After instruction: W = 01h C = 1 Z = 0
Example 2: **SUBLW 02H**	Before instruction: W = 02h After instruction: W = 00h C = 1 Z = 1
Example 3: **SUBLW 02H**	Before instruction: W = 03h After instruction: W = FFh C = 0 Z = 0

SUBWF	Subtract W from f
Syntax	SUBWF f, d 0 ≤ f ≤ FFh d ∈ [0,1]
Operation	[Destination] ← [f] – [W]
Flag Affected	C, DC, Z
Description	Subtract the content of the W register from [f]. If d is 0, the result is stored in the W register. If d is 1, the result is stored back in [f],
Cycle	1
Example 1: **SUBWF OPERAND, 1**	Before instruction: OPERAND = 33h, W = 01h After instruction: OPERAND = 32h C = 1 Z = 0
Example 2: **SUBWF OPERAND, 1**	Before instruction: OPERAND = 01h, W = 01h After instruction: OPERAND = 00h C = 1 Z = 1
Example 3: **SUBWF OPERAND, 1**	Before instruction: OPERAND = 04h, W = 05h After instruction: OPERAND = FFh C = 0 Z = 0

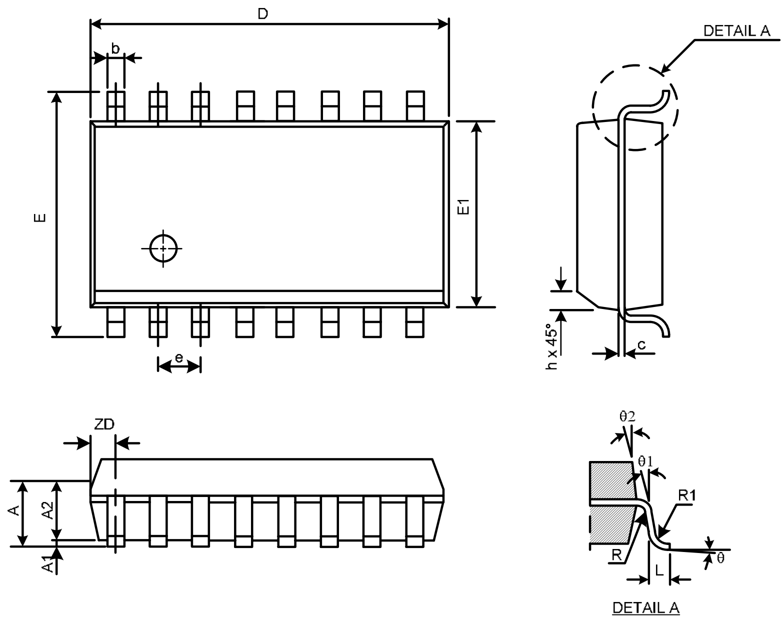
SUBWFC	Subtract W and Carry from f
Syntax	SUBWFC f, d 0 ≤ f ≤ FFh d ∈ [0,1]
Operation	[Destination] ← [f] – [W] –$\dot{C}$
Flag Affected	C, DC, Z
Description	Subtract the content of the W register from [f]. If d is 0, the result is stored in the W register. If d is 1, the result is stored back in [f].
Cycle	1
Example 1: **SUBWFC OPERAND, 1**	Before instruction: OPERAND = 33h, W = 01h C = 1 After instruction: OPERAND = 32h, C = 1, Z = 0
Example 2: **SUBWFC OPERAND, 1**	Before instruction: OPERAND = 02h, W = 01h C = 0 After instruction: OPERAND = 00h, C = 1, Z = 1
Example 3: **SUBWFC OPERAND, 1**	Before instruction: OPERAND = 04h, W = 05h C = 0 After instruction: OPERAND = FEh, C = 0, Z = 0

XORLW	Exclusive OR literal with W
Syntax	XORLW k 0 ≤ k ≤ FFh
Operation	[W] ← [W] XOR k
Flag Affected	Z
Description	Exclusive OR the content of the W register and the eight-bit literal "k". The result is stored in the W register.
Cycle	1
Example: **XORLW 5Fh**	Before instruction: W = ACh After instruction: W = F3h

XORWF	Exclusive OR W and f
Syntax	XORWF f, d 0 ≤ f ≤ FFh d ∈ [0,1]
Operation	[Destination] ← [W] XOR [f]
Flag Affected	Z
Description	Exclusive OR the content of the W register and [f]. If d is 0, the result is stored in the W register. If d is 1, the result is stored back in [f].
Cycle	1
Example: **XORWF OPERAND, 1**	Before instruction: OPERAND = 5Fh, W = ACh After instruction: OPERAND = F3h

12. Package Information

12.1 Package Outline & Dimensions



SYMBOL	DIMENSION IN MM			DIMENSION IN INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	1.35	1.63	1.75	0.053	0.064	0.069
A1	0.10	0.15	0.25	0.004	0.006	0.010
A2			1.50			0.059
b	0.20		0.30	0.008		0.012
c	0.18		0.25	0.007		0.010
e	0.635 BASIC			0.025 BASIC		
D	4.80	4.90	5.00	0.189	0.193	0.197
E	5.79	5.99	6.20	0.228	0.236	0.244
E1	3.81	3.91	3.99	0.150	0.154	0.157
L	0.41	0.635	1.27	0.016	0.025	0.050
h	0.25		0.50	0.010		0.020
L1	0.254 BASIC			0.010 BASIC		
ZD	0.229 REF			0.009 REF		
R1	0.20		0.33	0.008		0.013
R	0.20			0.008		
θ	0°		8°	0°		8°
θ1	0°		8°	0°		8°
θ2	5°	10°	15°	5°	10°	15°
JEDEC	MO-137(AB)					

Notes: Dimension D does not include mold protrusions or gate burrs.
Mold protrusions and gate burrs shall not exceed 0.06 inch per side.