

# **Implementing Triple Conversion Single-Phase On-line UPS using TMS320C240**

*Shamim Choudhury*
*DCS Applications*

## **Abstract**

Uninterruptible power supplies (UPS) play an important role in interfacing critical loads such as computers, communication systems, medical/life support systems, and industrial controls to the utility power grid. Among the various UPS topologies, on-line UPS provides the most protection to such loads against any utility power problem. However, because of the multiple power conversion stages, on-line UPSs have been the most complex and expensive type of system. Today's low-cost, high-performance digital signal processor (DSP) controllers, such as the Texas Instruments (TI™) TMS320C24x, provide an improved and cost-effective solution for on-line UPS design. This application report discusses the different implementation aspects of a DSP based on-line UPS design using a TMS320C240.

## **Contents**

Introduction .....	3
System Overview .....	4
Sampling Cycle .....	8
Output Inverter Stage.....	11
Inverter Controller Implementation .....	13
Output Inverter Voltage and Current Sensing.....	14
Battery Charger.....	17
Battery Charging Procedure .....	18
Battery Charger Control Algorithm.....	19
Battery Charger Controller Implementation .....	20
Battery Charger Voltage and Current Sensing .....	22
Input Power Factor Control (PFC).....	26
PFC Controller Implementation .....	28
Input PFC Voltage and Current Sensing.....	31
Battery Voltage Boost Stage .....	34
Boost Stage Controller Implementation .....	36
Boost Stage Voltage and Current Sensing .....	38
Software Organization.....	38
Experimental Results .....	43
Appendix A Example Code .....	45



## Figures

Figure 1.	Triple Conversion On-line UPS System.....	5
Figure 2.	Block Diagram of TMS320C240 Controlled On-line UPS System .....	6
Figure 3.	On-line UPS Topology .....	7
Figure 4.	Sampling Cycle in a 'C240-Based On-line UPS Design .....	8
Figure 5.	UPS Output Inverter Control.....	11
Figure 6.	Inverter Output Voltage ( $V_o$ ) Sense Amplifier.....	15
Figure 7.	Inverter Output Inductor Current ( $I_o$ ) Sense Amplifier .....	16
Figure 8.	UPS Battery Charger Control .....	17
Figure 9.	Trickle Charge Controller Diagram .....	18
Figure 10.	Over Charge Controller Diagram .....	19
Figure 11.	VB Sense Amplifier.....	23
Figure 12.	Lower DC Bus Capacitor Voltage ( $V_-$ ) Sense Amplifier .....	24
Figure 13.	Battery Inductor Current ( $I_b$ ) Sense Amplifier .....	25
Figure 14.	Input PFC Stage Implementation.....	26
Figure 15.	UPS Input Voltage ( $V_s$ ) Sense Amplifier.....	31
Figure 16.	Upper DC Bus Capacitor Voltage ( $V_+$ ) Sense Amplifier .....	32
Figure 17.	PFC Input Inductor Current ( $I_s$ ) Sense Amplifier .....	33
Figure 18.	UPS Battery Voltage Boost Stage Control.....	34
Figure 19.	Main Program Flowchart.....	38
Figure 20.	INT2 Interrupt Dispatcher Flowchart .....	39
Figure 21.	T1PINT Interrupt Service Routine Flowchart .....	40
Figure 22.	T1UFINT Interrupt Service Routine Flowchart.....	41
Figure 23.	INT3 Interrupt Dispatcher and Timer2 Underflow ISR Flowchart.....	42

## Tables

Table 1.	Control Loop Calculation Time for the 'C240-Based UPS Design .....	10
Table 2.	Inverter Output Voltage Scaling and Level Shifting .....	15
Table 3.	Inverter Output Inductor Current Scaling and Level Shifting.....	16
Table 4.	Voltage VB Scaling and Level Shifting.....	23
Table 5.	Lower DC Bus Capacitor Voltage ( $V_-$ ) Scaling and Level Shifting .....	24
Table 6.	Battery Inductor Current Scaling.....	25
Table 7.	UPS Input Voltage Scaling and Level Shifting .....	31
Table 8.	Upper DC Bus Capacitor Voltage ( $V_+$ ) Scaling.....	32
Table 9.	PFC Input Inductor Current Scaling and Level Shifting.....	33



## Introduction

Uninterruptible power supplies (UPS) play an important role in interfacing critical loads such as computers, communication systems, medical/life support systems, and industrial controls to the utility power grid. They are designed to provide clean and continuous power to the load under essentially any normal or abnormal utility power condition. Among the various UPS topologies or configurations, on-line UPS, also known as inverter-preferred UPS, offers the best line-conditioning performance and the most protection to the load against any utility power problems. It provides regulated sinusoidal output voltage under any input line condition. When powered from the utility power lines, it draws sinusoidal input current at a high input power factor. These improved input/output characteristics make on-line UPS the ideal solution in many applications. However, because of the use of multiple power conversion stages and the associated analog controllers, on-line UPS have traditionally been the most complex and expensive type of system. In addition to the analog controllers, on-line designs require the use of a low-end microcontroller to provide easy interface to a host computer in order to establish interactive communication and to implement adequate monitoring of the system. These multiple analog and digital controller based designs result in low component integration and increased system cost. High performance microcontrollers that can be used to achieve increased integration are available today, but they do not necessarily provide a cost effective solution.

Today's low-cost, high-performance DSP controllers, such as the Texas Instruments (TI™) TMS320C24x, provide an improved and cost effective solution for on-line UPS design. The 'C24x has integrated peripherals specifically chosen for embedded control applications. These include: analog-to-digital converters (ADCs), PWM outputs, timers, protection circuitry, serial communications, and other functions. High CPU bandwidth and the integrated power electronic peripherals of these devices make it possible to implement a complete digital control of on-line UPS. Most instructions for the 'C24x, including multiplication and accumulation (MAC) as one instruction, are single cycle. Therefore, multiple control algorithms can be executed at high speed, making it possible to achieve the required high sampling rate for good dynamic response. This also makes it possible to implement multiple control loops of an on-line UPS in a single chip.

This results in increased integration and lower system cost. Digital control also brings the advantages of programmability, immunity to noise, and eliminates redundant voltage and current sensors for each controller. With fewer components, the system requires less engineering time, and it can be made smaller and more reliable. DSP control offers another big advantage over traditional analog control -- software. The extra DSP bandwidth is available for implementing more sophisticated algorithms, as well as communications to host systems and I/O devices such as LCD displays. DSP programmability means that it is easy to update systems with enhanced algorithms for improved reliability.

For all these reasons, the 'C24x provides an ideal solution for on-line UPS design.



This application report discusses the different implementation aspects of a DSP-based on-line UPS design. The DSP that is used is the TMS320C240. The major features of the TMS320C240 that are useful for on-line UPS design include:

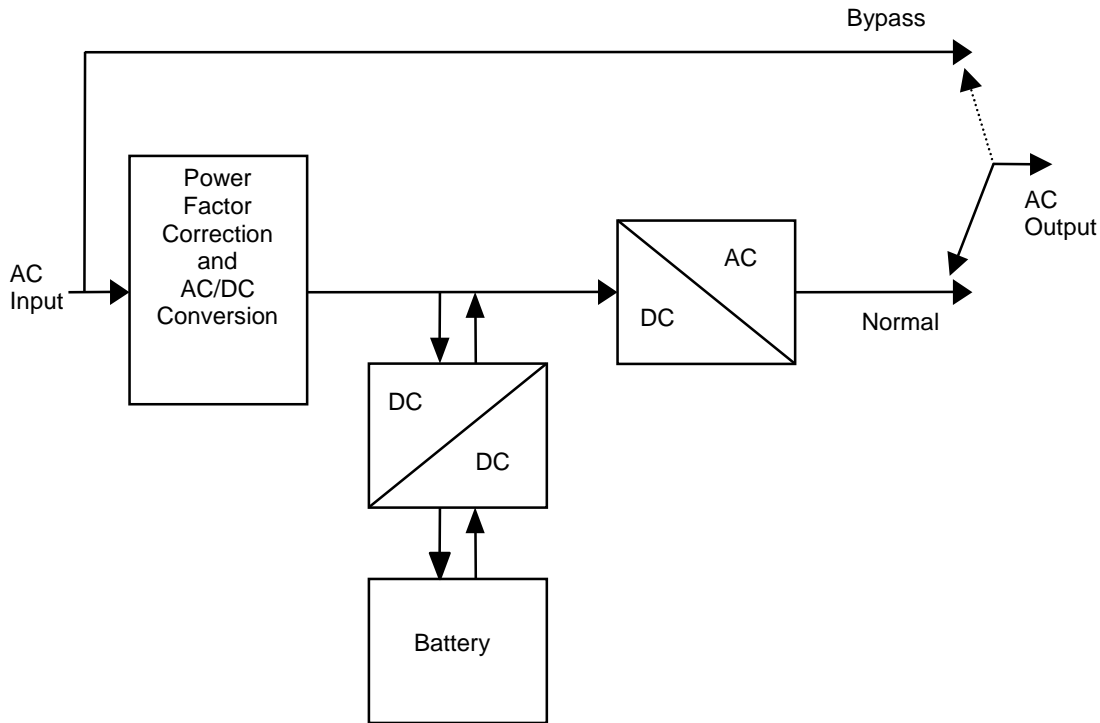
- ❑ TMS320C2xx CPU core with 50ns instruction cycle time
- ❑ 544 words of on-chip data/program memory, 16K words of on-chip program ROM or Flash EEPROM, 64K words of program, 64K words of data, and 64K words of I/O address space
- ❑ Dual 10-bit ADC with 8 $\mu$ s of converter time per two input channels; 8 analog inputs for each ADC module, totaling 16 analog inputs
- ❑ PLL, watchdog timer, SCI, SPI, and 28 multiplexed I/O pins
- ❑ 12 compare/PWM outputs, 9 that are independent
- ❑ Three general-purpose up and up/down timers, each with a 16-bit compare unit capable of generating one independent PWM output
- ❑ Three 16-bit simple compare units capable of generating 3 independent PWM outputs
- ❑ Power drive protection (PDPINT) input for the safe operation of power converters.
- ❑ Six maskable core interrupts, 3 of which accept 'C240's event manager (EM) interrupts from 23 different sources.

The 'C240 has all the necessary features for implementing a highly integrated on-line UPS design.

## System Overview

A triple conversion on-line UPS system is shown in Figure 1. The power factor correction (PFC) input stage is an ac-to-dc converter, which rectifies the input  $V_{ac}$  and creates the dc bus voltage while maintaining sinusoidal input current at a high input power factor. The PFC stage also regulates the dc bus voltage against variation in input  $V_{ac}$ . The dc bus voltage is inverted through the output dc-to-ac inverter stage to generate the output  $V_{ac}$  of appropriate frequency. A dc-to-dc buck converter stage implements the battery charger. The battery charger stage steps down the high dc bus voltage (up to 400 V) to allow a smaller battery to be charged. A dc-to-dc boost converter raises the battery voltage up to the bus voltage when the system is operating in battery backup mode.

Figure 1. Triple Conversion On-line UPS System

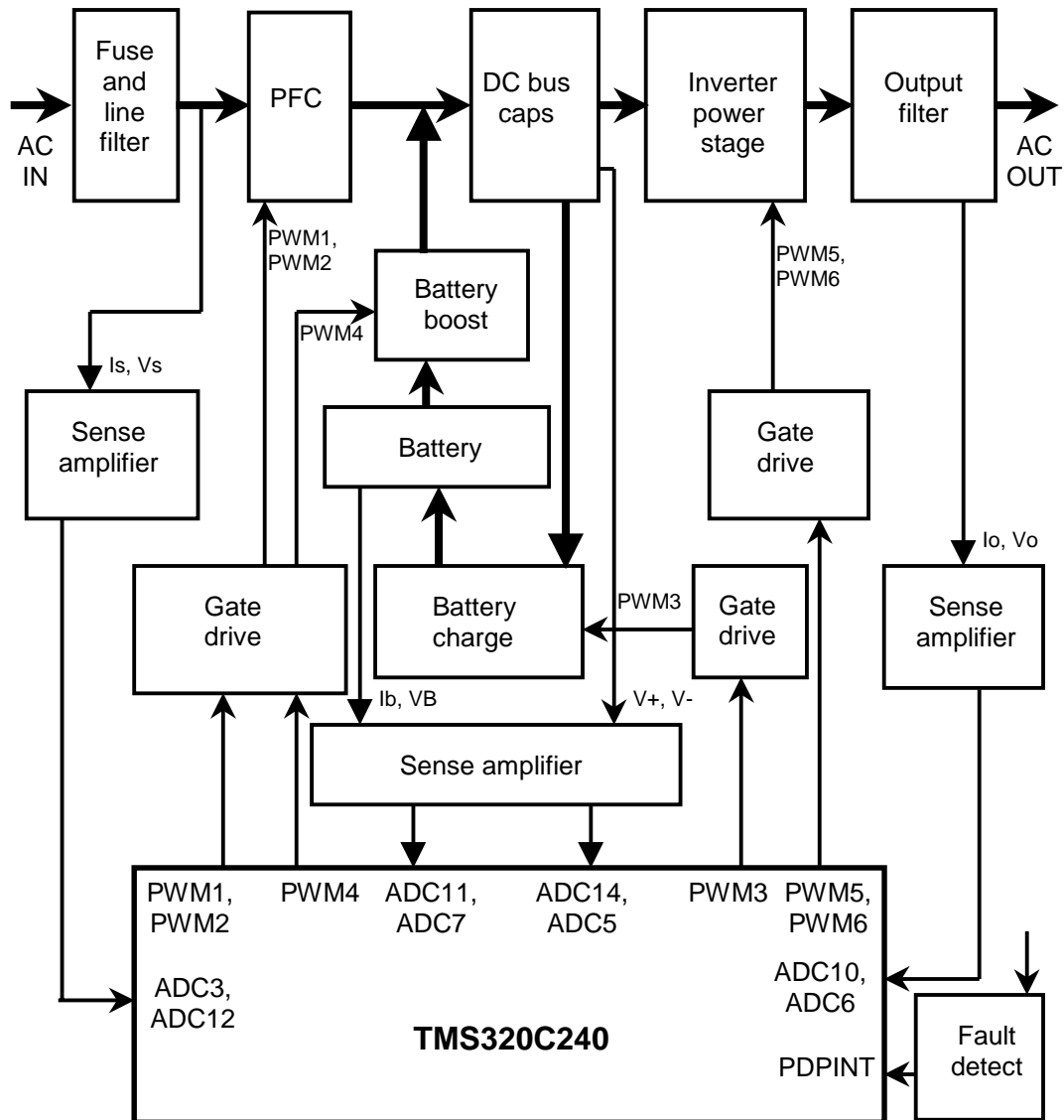


A triple conversion on-line UPS system has two operating modes. Under normal conditions, when ac input power is available, the input PFC stage, the battery charger, and the output inverter operate simultaneously. But when there is an input power failure, the battery supplies output power. During an input power failure, the battery voltage boost stage and the output inverter operate simultaneously to maintain the output.

Figure 2 shows a block diagram of the implementation of a triple-conversion on-line UPS system based on the 'C240. Four power stages, the input PFC stage, the output inverter stage, the battery charger stage, and the battery voltage boost stage, are all controlled by a single 'C240. Each power converter stage is a control system by itself and is characterized by double control loops, an inner current loop and an outer voltage loop. The bandwidth of the 'C240 makes it possible to implement these control loops in a single chip. With a performance rating of 20 MIPS, the 'C240 can handle the current and voltage control loops within the required real-time constraints. In addition, the device integrates the peripherals that are needed for UPS embedded control.

Eight signal samples are required to implement closed loop control of the four power converter stages. These are input  $V_{ac}$ , input inductor current, dc bus capacitor voltages (upper and lower capacitor voltages for voltage doubler configuration), output voltage, output inductor current, battery terminal voltage, and battery inductor current. As shown in Figure 2, eight integrated ADC channels of the 'C240 sample these voltage and current signals. With these signal samples, the CPU implements the desired control algorithms for multiple power stages and calculates the required PWM duty cycles. This would be analogous to feedback and error amplifier compensation circuits in analog control.

Figure 2. Block Diagram of TMS320C240 Controlled On-line UPS System



These calculated duty cycle values are then used in the integrated PWM modules to generate six PWM outputs to control the power stage switches. A programmable dead time prevents any short circuit condition across the dc bus capacitor. The dead time can be programmed so that the ON states of the two switches do not overlap. The PDPINT input of the 'C240 is used to shut down the power stages in the event of an overcurrent or short-circuit condition.

Figure 3 shows the triple-conversion topology of this 'C240-based on-line UPS design. The major modules of the design are the input PFC stage, the battery voltage boost stage, the battery charger stage, and the output inverter stage. One thing that should be noted is the common neutral feature; that is, the output and input have a common neutral, which is required by regulation without a transformer.

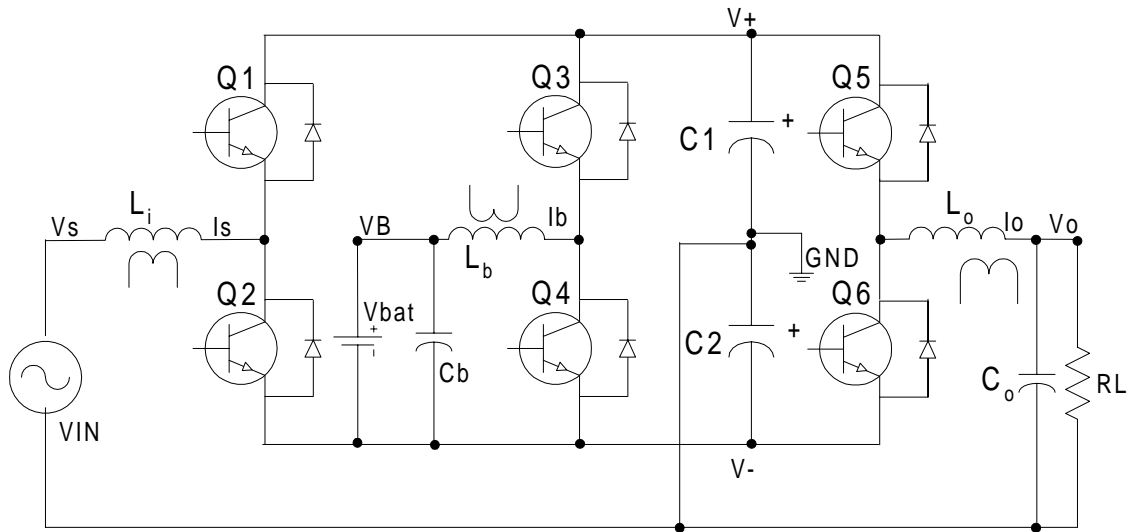
The input PFC stage consists of power devices Q1 and Q2, inductor  $L_i$ , and bus capacitors C1 and C2. It provides input power factor correction and boosts the bus voltage to 400 Vdc.

The output inverter stage consists of bus capacitors C1 and C2, power devices Q5 and Q6, output inductor  $L_o$ , and capacitor  $C_o$ . It generates a sine wave output voltage.

The battery voltage boost stage consists of power device Q4, inductor  $L_b$ , and bus capacitors C1 and C2. It operates like a typical dc/dc boost converter and engages only when the UPS operates on the battery to boost the battery voltage from 110 Vdc to a bus voltage of 400 Vdc.

The battery charger consists of power device Q3 and inductor  $L_b$ . This is basically a dc/dc buck converter, which allows charging of the 110 Vdc battery from the 400 Vdc bus.

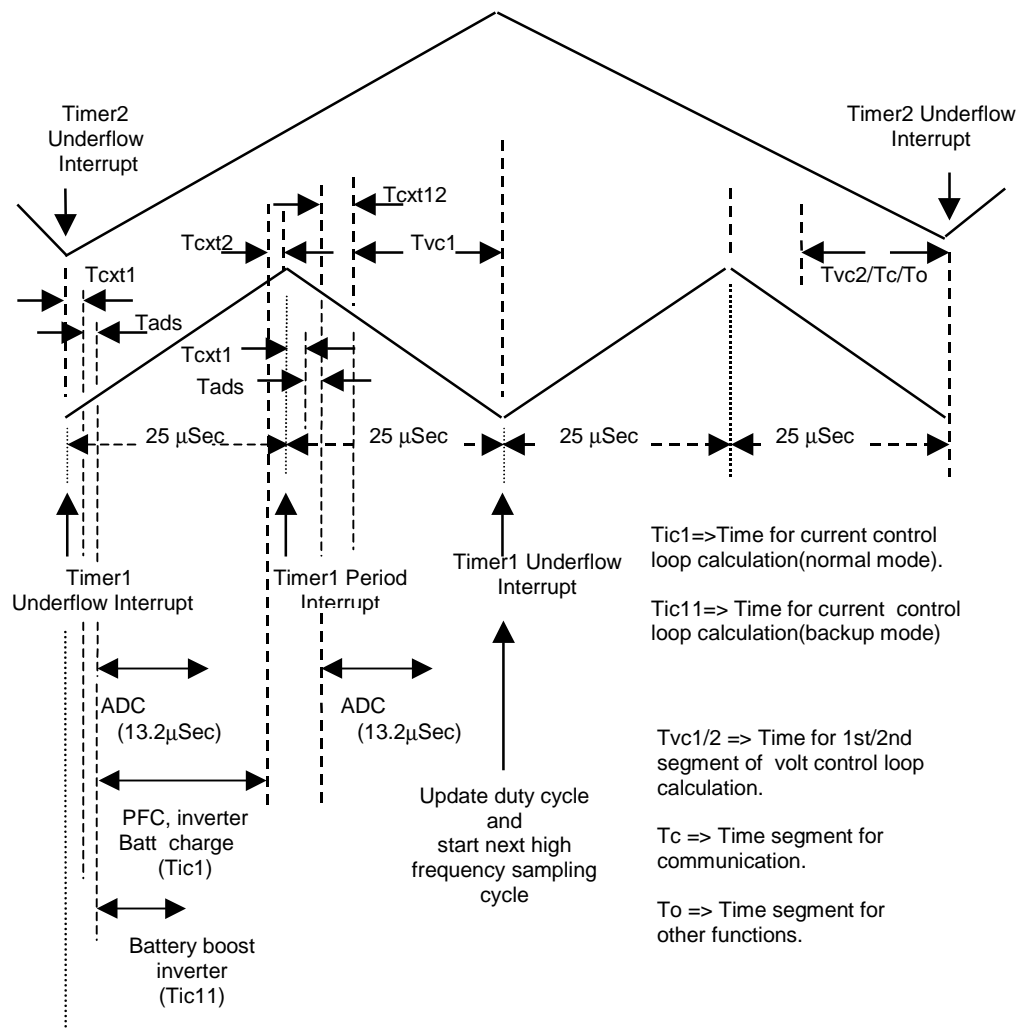
Figure 3. On-line UPS Topology



## Sampling Cycle

Figure 4 shows the control loop sampling cycle for the on-line UPS implementation based on the 'C240. Two general-purpose timers of the 'C240, GP Timer1 and GP Timer2, are used to implement the sampling loops. GP Timer1 provides the time base for PWM generation, ADC sampling, and high frequency current control loops. GP Timer2 is the time base for the low frequency voltage control loops. Both timers operate in continuous up/down counting mode. For this design, the current loop sampling frequency is 20kHz and the voltage loop sampling frequency is 10kHz. As shown in Figure 4, the sampling time for the faster current loop is 50 $\mu$ s, and the sampling time for the slower voltage loop is 100 $\mu$ s. Interrupt mask registers IMR, EVIMRA, and EVIMRB are configured to allow Timer1 to generate an interrupt on underflow and period match, and Timer2 to generate an interrupt only on underflow.

Figure 4. Sampling Cycle in a 'C240-Based On-line UPS Design







As shown in Figure 4, both Timer1 underflow (T1UF) interrupt and Timer2 underflow (T2UF) interrupt occur at the same time. T1UF interrupt is serviced first, because it generates an interrupt request to the core on INT2 and, therefore, has a higher priority to T2UF interrupt (INT3 level). Once the core receives the INT2 interrupt, it takes a finite amount of time,  $T_{cxt1}$ , for interrupt source identification and context saving. Following that, in the T1UF interrupt service routine (ISR), ADC data registers are read and the conversion results from the previous four conversions are saved. Then ADC control registers are configured for starting four new conversions. Saving the ADC results and starting new conversions require a finite time,  $T_{ads}$ . Once the ADC conversion starts, the conversion process and the current control loop calculation run in parallel. The time required for current control loop calculation is indicated as  $T_{ic1}$  in normal mode, and as  $T_{ic11}$  in backup mode. Once these operations are complete, a finite time,  $T_{cxt2}$ , is required to restore the context before the program returns from this T1UF ISR. After exiting this T1UF ISR, the core acknowledges another interrupt on INT2, this time the source being the Timer1 period (T1PR) match. Therefore, the servicing of the pending T2UF interrupt on INT3 is delayed once again and the core responds to INT2 to service T1PR interrupt. In responding to this INT2 interrupt, again a finite time,  $T_{cxt1}$ , is needed for interrupt source identification and context saving. Following that, in the T1PR ISR, ADC data registers are read and the conversion results from the previous four conversions are saved. Then ADC control registers are configured for starting four new conversions. Saving the ADC results and starting the new conversions require a finite time,  $T_{ads}$ . Once the ADC conversion starts, the program restores the saved context and returns from this T1PR ISR. After exiting this T1PR ISR, the core services the pending T2UF interrupt on INT3. Again a finite time is required to identify the interrupt source and to save the context. This time plus the time required for restoring context of T1PR ISR is indicated as  $T_{cxt12}$  in Figure 4. Once this is done, T2UF ISR is serviced and the voltage control loops are calculated. The time spent in calculating the voltage control loops is  $T_{vc1}$ . If this calculation does not complete before the next T1UF interrupt occurs, T2UF interrupt is interrupted by this T1UF interrupt. When this happens, the remaining portion of the voltage control loop is calculated after servicing the next T1UF and T1PR interrupts. This time is shown as  $T_{vc2}$ . Once the voltage control loop calculation is complete, the 'C240 bandwidth allows the implementation of interactive communication and other functions using the remaining time. These time segments are indicated as  $T_c$  and  $T_o$ .

In this design, Timer1 underflow causes the full compare registers (CMPRx,  $x=1,2,3$ ) to update with the values in their respective shadow registers. This changes the duty cycle of the PWM outputs. The duty cycle values in the shadow registers are based on the calculation performed in the previous high-frequency sampling cycle.

Timer1 underflow interrupts the CPU and the program branches to the corresponding interrupt service routine, where it performs the following tasks:

- 1) Reads the four ADC samples from the two conversions previously started by Timer1 period interrupt, since Timer1 period and underflow interrupts are generated alternately. These signals are output voltage  $V_o$ , output inductor current  $I_o$ , input voltage  $V_s$ , and input inductor current  $I_s$ .
- 2) Starts both ADCs for a new conversion followed by a second one. Since the ADC channel selector bits and the start of conversion bit in the ADC control register 1 (ADCTRL1) are shadowed, these bits can be reconfigured for a second conversion while the first conversion is still going. The effect of writing to these bits occurs after the first conversion finishes. Configuring ADCTRL1 in this way, allows conversion of four signals from two back-to-back conversions by both the ADC modules. The signals for which ADC conversions are started are upper dc bus capacitor voltage



V+, lower dc bus capacitor voltage V-, battery voltage VB, and battery inductor current Ib.

- 3) Executes the algorithm that detects the positive and negative half cycle of the input voltage Vs. This information is required for PFC stage implementation.
- 4) Executes the current control algorithms, calculates the PWM duty cycles and saves the values in the specified full compare registers (CMPRx, x=1,2,3). Since the compare registers are shadowed, these values go into the respective shadow registers. As mentioned earlier, the compare registers are updated with these values in the shadow registers when the next Timer1 underflow occurs.
- 5) Returns from the interrupt service routine.

Timer1 period match interrupts the CPU and the program branches to the corresponding interrupt service routine, where it performs the following tasks:

- 1) Reads the four ADC samples from the two conversions previously started by Timer1 underflow interrupt. These signals are upper dc bus capacitor voltage V+, lower dc bus capacitor voltage V-, battery voltage VB, and battery inductor current Ib.
- 2) Starts both ADCs for a new conversion followed by a second one. This allows conversion of four signals. As mentioned earlier, this is possible because of the shadowed bits in ADCTRL1. The signals for which ADC conversions are started here are output voltage Vo, output inductor current Io, input voltage Vs, and input inductor current Is.
- 3) Returns from the interrupt service routine.

Timer2 underflow interrupts the CPU and the program branches to the corresponding interrupt service routine, where it performs the following tasks:

- 1) Executes the sine wave generation program to generate the reference sine wave for the output inverter stage.
- 2) Executes the voltage control algorithms to generate the current commands for the corresponding high frequency current control loops.
- 3) Returns from the interrupt service routine.

Timer2 underflow interrupt is made interruptible by Timer1 interrupts. Therefore, the voltage control algorithms are executed only when the Timer1 interrupt service routines are not being executed.

Table 1 shows the control loop calculation time for the 'C240-based UPS design.

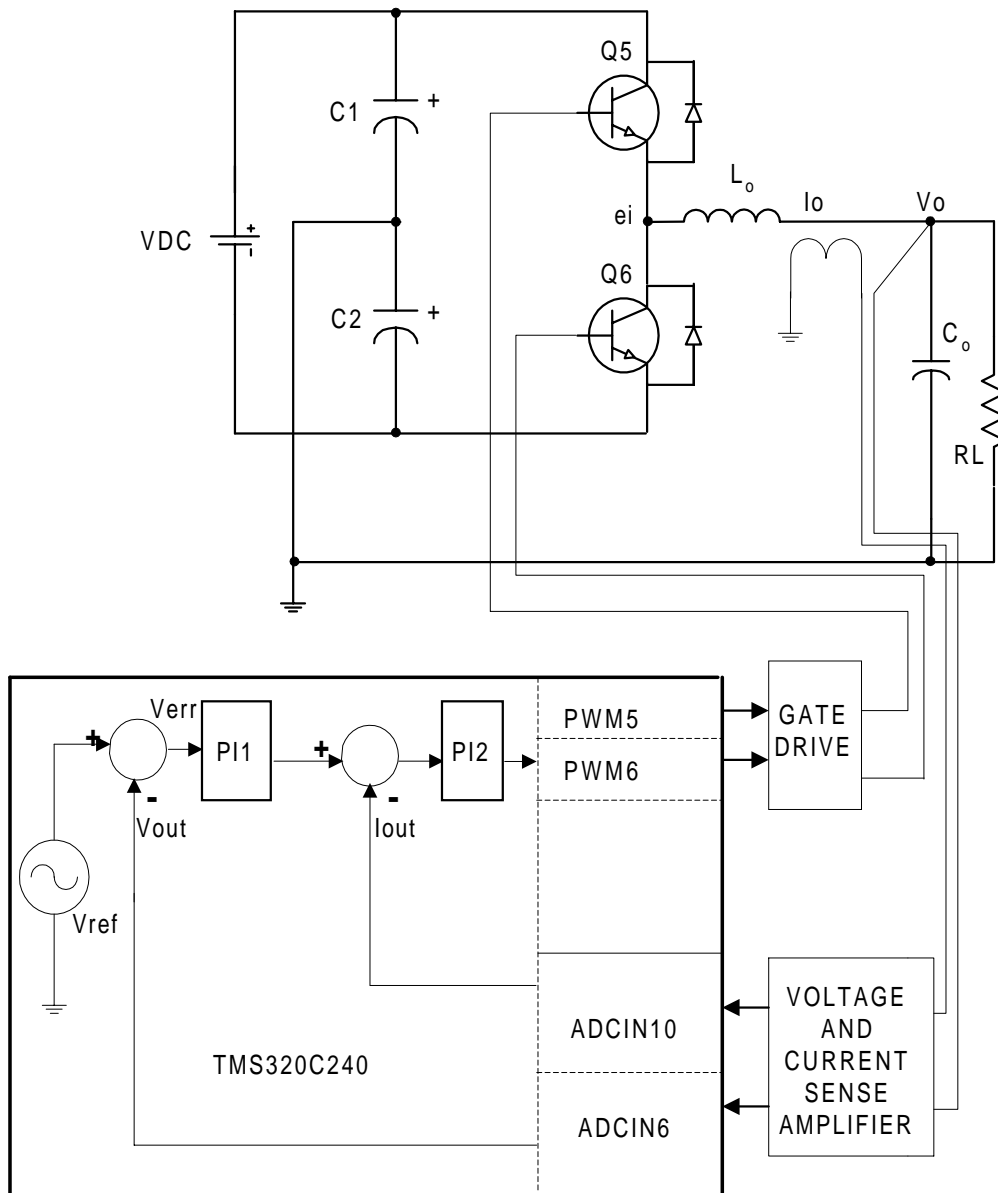
*Table 1. Control Loop Calculation Time for the 'C240-Based UPS Design*

Mode	Operations	Time (μs)
Normal mode	Current control loop calculation	Tic1 = 22
Normal mode	Voltage control loop calculation	Tvc1 + Tvc2 = 17
Backup mode	Current control loop calculation	Tic11 = 10.5
Backup mode	Voltage control loop calculation	Tvc1 = 12

## Output Inverter Stage

Figure 5 shows the single phase UPS output inverter stage interfaced to the 'C240. As indicated in the figure by the block labeled TMS320C240, the DSP implements all the necessary control functions for this stage.

Figure 5. UPS Output Inverter Control





The inverter modulates a dc bus voltage, Vdc, into a cycle-by-cycle average output voltage. The amplitude is directly proportional to the commanded duty cycle of the inverter and the amplitude of the dc bus voltage Vdc. It can range from + Vdc to - Vdc. Current mode control is used for this PWM inverter. Current mode control is a two-loop control system that simplifies the design of the outer voltage control loop and improves UPS performance in many ways, including better dynamics and a feed forward characteristic that could be used to compensate DC bus ripple and dead-time effect, etc.

The system parameters used in this design are the following:

$$V_{dc} = 400V$$

$$f_s = 20kHz$$

$$L_o = 300\mu h$$

$$C_o = 20\mu f$$

$$R_c = 0.02 \text{ ohm}$$

$$R_L = 14.4 \text{ ohms}$$

For this system, the current loop compensation is:

$$G_{I\_INV}(s) = 2.00 \times \frac{1 + 1.061 \times 10^{-4} s}{1.061 \times 10^{-4} s} \quad (1)$$

and the voltage loop compensation is:

$$G_{V\_INV}(s) = 1.00 \times \frac{1 + 3.183 \times 10^{-4} s}{3.183 \times 10^{-4} s} \quad (2)$$

As shown in Figure 5, the instantaneous inverter output voltage Vo and the inductor current Io are sensed and conditioned by the respective voltage and current sense amplifiers. This block is labeled as VOLTAGE AND CURRENT SENSE AMPLIFIER and is explained in the *Output Inverter Voltage and Current Sensing* section. The sensed voltage and current signals are then fed back to the DSP by the two ADC channels ADCIN06 and ADCIN10, respectively. The digitized feedback output voltage, Vout, is compared to an internally generated sine wave reference Vref. The difference between these two voltages, Verr, is fed into the PI regulator PI1, which is based on GV\_INV(s) in Equation (2). The output of this compensator is the reference current command for the inner current loop. This reference is compared with the digitized inductor current feedback Iout and then the difference is passed to the second PI regulator, PI2, based on GI\_INV(s) in Equation (1). The output of this current regulator is the command voltage, which is used to determine the duty cycle of the PWM gating signals. This current regulator output is first converted to a proportional Q0 number and then passed onto the PWM module through the full compare register CMPR3. The PWM module compares this value with a 20kHz triangle waveform generated internally by Timer1. The result of this comparison is the required PWM signals PWM5 and PWM6, which control the switches Q5 and Q6, respectively. Programmable precise dead times are automatically provided between this pair of complementary PWM signals. This dead time is defined by the dead-time control register DBTCON.

For this design, the current loop sampling frequency is 20kHz, that is, the sampling time of the inductor current is 50μs. The sampling time of output voltage is 100μs corresponding to a voltage loop sampling frequency of 10kHz.

## Inverter Controller Implementation

The PI controller in Equation (1) is transformed to an equivalent digital form, as shown below, before being implemented by 'C240:

$$G_{I\_INV}(s) = 2.00 \times \frac{1 + 1.061 \times 10^{-4} s}{1.061 \times 10^{-4} s}$$

$$\Rightarrow G_{I\_INV}(s) = K_p + \frac{K_I}{s} = \frac{U(s)}{E(s)}$$

Where,

$$K_p = 2$$

$$K_I = 18849.556$$

In discrete form,

$$U(n) = K_p E(n) + K_I T_{SI} \sum_{i=0}^n E(i)$$

where the current loop sampling time is,

$$T_{SI} = 50 \times 10^{-6} \text{ seconds}$$

This is implemented with output saturation and integral component correction using the following three equations:

$$U(n) = K0_{inv} * E(n) + I(n)$$

$$I(n) = I(n-1) + K1_{inv} * E(n) + Kcorr_{inv} * Epi_{inv}$$

$$Epi_{inv} = U(n) - Us$$

where,

$$U(n) \geq U_{\max} \Rightarrow Us = U_{\max}$$

$$U(n) \leq U_{\min} \Rightarrow Us = U_{\min}$$

otherwise,

$$Us = U(n)$$

The coefficients are defined as,

$$K0_{inv} = K_p = 2 = 400h(q9),$$

$$K1_{inv} = K_I T_{SI} = 0.942 = 1E2h(q9)$$

$$Kcorr_{inv} = \frac{K1_{inv}}{K0_{inv}} = 0.471 = F12h(q13)$$

In a similar manner, the PI controller in Equation (2) is transformed to an equivalent digital form, as shown below, before being implemented by 'C240:



$$G_{V\_INV}(s) = K_p + \frac{K_I}{s} = \frac{U(s)}{E(s)}$$

Where,

$$K_p = 1,$$

$$K_I = 3141.59$$

Knowing the voltage loop sampling frequency of 10kHz, this is implemented with output saturation and integral component correction in the following form:

$$U(n) = K0_{vinv} * E(n) + I(n)$$

$$I(n) = I(n-1) + K1_{vinv} * E(n) + Kcorr_{vinv} * Epi_{vinv}$$

$$Epi_{vinv} = U(n) - Us$$

where,

$$U(n) \geq U_{max} \Rightarrow Us = U_{max}$$

$$U(n) \leq U_{min} \Rightarrow Us = U_{min}$$

otherwise,

$$Us = U(n)$$

The coefficients are,

$$K0_{vinv} = K_p = 1 = 7FFFh(q15),$$

$$K1_{vinv} = K_I T_{SV} = 0.31416 = 507h(q12)$$

$$Kcorr_{vinv} = \frac{K1_{vinv}}{K0_{vinv}} = 0.31416 = 507h(q12)$$

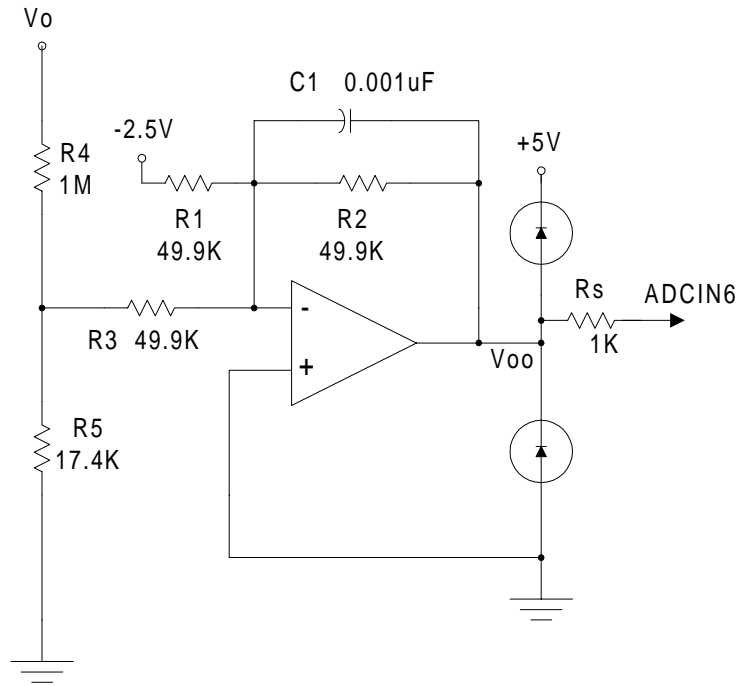
$$T_{SV} = 100 \times 10^{-6} \text{ seconds}$$

## Output Inverter Voltage and Current Sensing

As shown in Figure 5, the instantaneous inverter output voltage  $V_o$  and the inductor current  $I_o$  are first sensed and conditioned by the voltage and current sense amplifiers before being applied to the 'C240.

Inverter output voltage ( $V_o$ ) is scaled and level shifted to bring the voltage into the range of ADC by using the output voltage sense amplifier circuit shown in Figure 6.

Figure 6. Inverter Output Voltage (Vo) Sense Amplifier



For the amplifier circuit shown in Figure 6, the input voltage ( $V_{oo}$ ) to ADC channel ADCIN6 is calculated by the equation:

$$V_{oo} = 2.5 - \frac{V_o(17400 \parallel 49900)}{10^6 + (17400 \parallel 49900)}$$

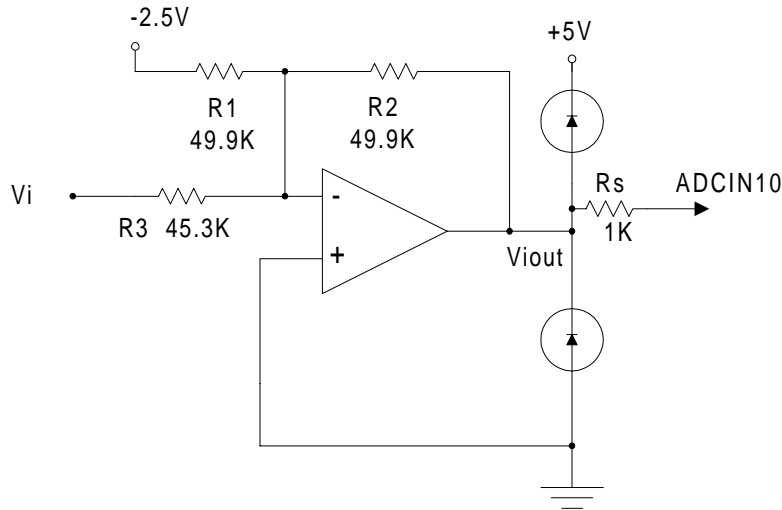
The inverter output voltage ( $V_o$ ), the corresponding input voltage ( $V_{oo}$ ) to ADC channel ADCIN6, and the resulting ADC data register values are listed in Table 2.

Table 2. Inverter Output Voltage Scaling and Level Shifting

$V_o$ (V, peak)	$V_{oo}$ (Vdc)	ADC_FIFO
+196	0	0000h
0	2.5	7FC0h
-196	5	FFC0h

Inverter output inductor current ( $I_o$ ) is sensed by the current sensor. This current sensor output voltage ( $V_i$ ) is scaled and level shifted to bring the voltage into the range of ADC by using the output inductor current sense amplifier circuit shown in Figure 7. The current sensor used in this design generates 0.16 V per ampere of current.

Figure 7. Inverter Output Inductor Current ( $I_o$ ) Sense Amplifier



For the amplifier circuit shown in Figure 7, the input voltage ( $V_{iout}$ ) to ADC channel ADCIN10 is calculated by the equation:

$$V_{iout} = 2.5 - \frac{49900V_i}{45300}$$

The inverter output inductor current ( $I_o$ ), the corresponding sensor output voltage ( $V_i$ ), the input voltage ( $V_{iout}$ ) to ADC channel ADCIN10, and the resulting ADC data register values are listed in Table 3.

Table 3. Inverter Output Inductor Current Scaling and Level Shifting

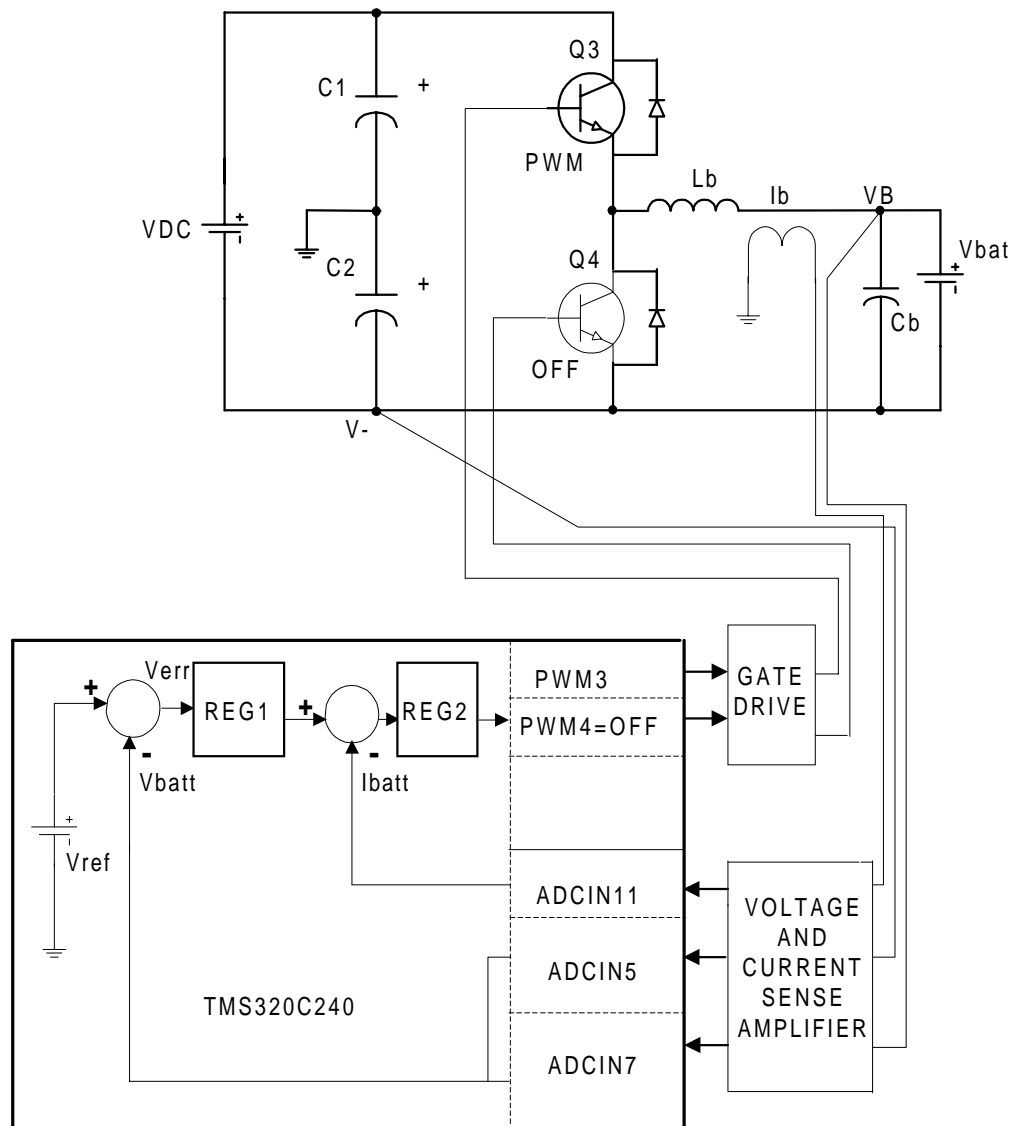
$I_o$ (A)	$V_i$ (V)	$V_{iout}$ (V)	ADCFIFO
+14.2	2.27	0	0000h
0	0	2.5	7FC0h
-14.2	-2.27	5	FFC0h



## Battery Charger

Figure 8 shows the battery charger stage interfaced to the 'C240. The charger is composed of power device Q3 and inductor  $L_b$ . This is basically a dc/dc buck converter, which allows charging of the 110 Vdc battery from the 400 Vdc bus. Two signals are required to implement the control algorithm: the battery inductor current  $I_b$  and the battery terminal voltage  $V_{bat}$ . The  $V_{bat}$  is measured indirectly by measuring two voltages,  $V_B$  and  $V_-$ , and then calculating  $V_{bat}$  from them.  $V_B$  is the battery positive terminal voltage with respect to GND and  $V_-$  is the capacitor C2 negative terminal voltage with respect to GND.

Figure 8. UPS Battery Charger Control

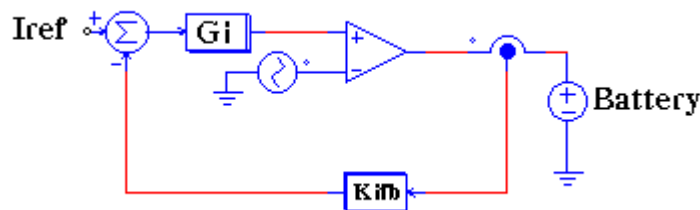


## Battery Charging Procedure

### Phase 1: Trickle Charge Mode

If the battery open circuit cell voltage is less than 1.80V before the charging starts, the charging procedure begins with trickle charge. In the trickle charging mode, the charging current is regulated at  $C/100$ . The operation remains in trickle mode until OCV is equal to or greater than 1.80V. The controller diagram is illustrated in Figure 9.

Figure 9. Trickle Charge Controller Diagram



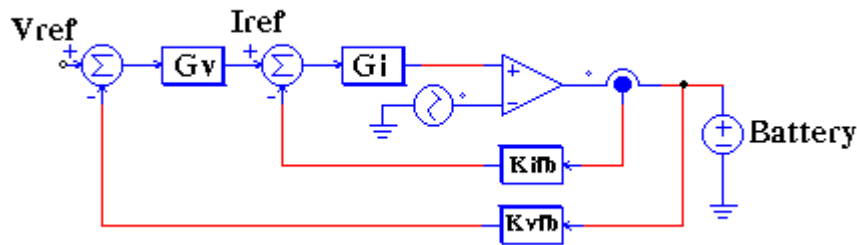
### Phase 2: Bulk Charge Mode

As soon as the cell VOC exceeds 1.80V, the operation turns into bulk charge mode. In the bulk charge mode, the charging current is regulated at  $C/3$ . In this particular unit, the current shall not exceed 2A, whatever C is of the battery. It is a constant current control and remains a constant charging current until the battery cell open circuit voltage reaches 2.40V. During this period, the controller diagram is similar to Figure 9, except the charging current reference is increased.

### Phase 3: Over Charge Mode

When the cell open circuit voltage reaches 2.40V, the operation becomes over charge mode. At this time, the charge control switches from constant charging current to constant charging voltage. The battery cell voltage is regulated at 2.40 V, while the charging current is tapped down continuously. The over charge mode lasts until the charging current is tapped down to less than  $C/12$ . The controller diagram during the over charger period is illustrated in Figure 10.

Figure 10. Over Charge Controller Diagram



For this design, only the overcharge mode is implemented, since this involves the implementation of two control loops and therefore, requires the maximum CPU time.

## Battery Charger Control Algorithm

As shown in Figure 8, the instantaneous voltages  $V_B$  and  $V_-$ , and the inductor current  $I_b$  are first sensed and conditioned by the respective sense amplifier circuits in the block labeled VOLTAGE AND CURRENT SENSE AMPLIFIER. This block is further explained in the *Battery Charger Voltage and Current Sensing* section. The sensed signals  $V_B$ ,  $V_-$ , and  $I_b$  are then fed back to DSP by the three ADC channels ADCIN7, ADCIN5, and ADCIN11, respectively.

The battery terminal voltage is calculated from the two signals,  $V_B$  and  $V_-$ , as:

$$V_{batt} = V_B - (V_-)$$

This calculated voltage,  $V_{batt}$ , is compared to the desired reference battery voltage  $V_{ref}$ . The difference between these two voltages is fed into the voltage controller REG1 based on  $G_V(s)$ , where,

$$G_V(s) = \frac{454(s + 500)}{s(s + 25132)}$$

The output of this controller is the reference charging current command for the inner current loop. This reference current is compared with battery inductor current feedback  $I_{batt}$  and then the difference is passed to the current controller REG2 based on  $G_I(s)$ , where,

$$G_I(s) = \frac{2123(s + 35714)}{s(s + 173720)}$$



The output of this current controller is the command voltage, which is used to determine the duty cycle of the PWM gating signal. The current regulator output is first converted to a proportional Q0 number and then passed onto the PWM module through the full compare register CMPR2. The PWM module compares this value with a 20kHz triangle waveform generated internally by Timer1. This generates the required PWM signal PWM3, which controls the switch Q3. During the charger operation Q3 is in PWM mode and Q4 is turned off. This is accomplished by configuring the action control register ACTR for active high PWM3 and forced low PWM4.

For this design, the sampling frequency for the battery current is 20kHz, and that for the battery voltage is 10kHz.

## Battery Charger Controller Implementation

The current controller is transformed to the equivalent digital form, as shown below, before being implemented by the 'C240.

Beginning with the analog controller:

$$G_I(s) = \frac{K_I(s+a)}{s(s+b)}$$

where,  $K_I = 2123, a = 35714, b = 173720$

Using bilinear transformation:

$$s = \frac{2(z-1)}{T(z+1)},$$

and substituting

$$G_I(s) = \frac{U_I(s)}{E_I(s)},$$

the current controller can be expressed as,

$$\frac{U_I(z)}{E_I(z)} = \frac{K_0 + K_1 z^{-1} + K_2 z^{-2}}{1 - K_3 z^{-1} - K_4 z^{-2}}$$

From this the final form of the digital current controller is,

$$U_I(n) = K_0 E_I(n) + K_1 E_I(n-1) + K_2 E_I(n-2) + K_3 U_I(n-1) + K_4 U_I(n-2)$$

where,

$$K_0 = \frac{K_I T(2 + aT)}{2(2 + bT)} = 0.018803 = 134h \text{ (Q14),}$$

$$K_1 = \frac{K_I T(2aT)}{2(2 + bT)} = 0.017738 = 123h \text{ (Q14),}$$

$$K_2 = \frac{K_I T(aT - 2)}{2(2 + bT)} = -1.06438 \times 10^{-3} = FFEFh \text{ (Q14),}$$

$$K_3 = \frac{4}{2 + bT} = 0.37432 = 17F5h \text{ (Q14),}$$

$$K_4 = \frac{bT - 2}{bT + 2} = 0.625678 = 5016h \text{ (Q15),}$$

$$T = 50 \times 10^{-6} \text{ seconds}$$

In a similar manner, the voltage controller is transformed to the form as shown below before being implemented by the 'C240.

The analog voltage controller:

$$G_V(s) = \frac{K_V(s + a)}{s(s + b)}$$

where,  $K_V = 454, a = 500, b = 25132$

Using bilinear transformation and substituting

$$G_V(s) = \frac{U_V(s)}{E_V(s)},$$

the voltage controller can be expressed as,

$$\frac{U_V(z)}{E_V(z)} = \frac{K_0 + K_1 z^{-1} - K_2 z^{-2}}{1 - K_3 z^{-1} - K_4 z^{-2}}$$

Therefore, the final form of the digital voltage controller is,

$$U_V(n) = K[k_0 E_V(n) + k_1 E_V(n-1) - k_2 E_V(n-2)] + K_3 U_V(n-1) + K_4 U_V(n-2)$$



where,

$$K = K_1 = \frac{K_v T(2aT)}{2(2 + bT)} = 5.044 \times 10^{-4} = 421Dh \text{ (Q25)},$$

$$k_0 = \frac{K_0}{K} = \frac{K_v T(2 + aT)}{2(2 + bT)K} = 20.4996 = 520h \text{ (Q6)},$$

$$k_1 = \frac{K_1}{K} = \frac{K_v T(2aT)}{2(2 + bT)K} = 1.0 = 3Fh \text{ (Q6)},$$

$$k_2 = \frac{K_2}{K} = \frac{K_v T(2 - aT)}{2(2 + bT)K} = 19.5017 = 4E0h \text{ (Q6)},$$

$$K_3 = \frac{4}{(2 + bT)} = 0.8888 = 38E2h \text{ (Q14)},$$

$$K_4 = \frac{bT - 2}{bT + 2} = 0.1111 = 1C72h \text{ (Q16)},$$

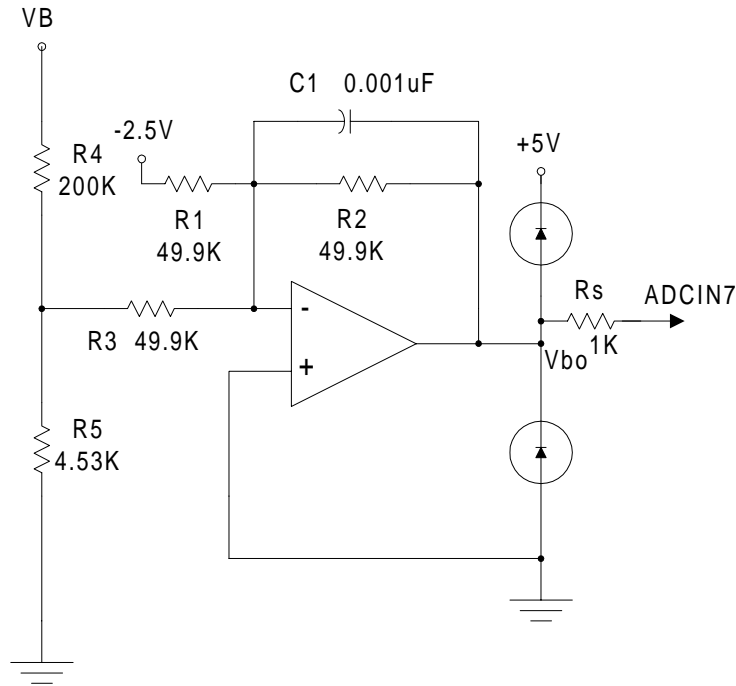
$$T = 100 \times 10^{-6} \text{ seconds}$$

## Battery Charger Voltage and Current Sensing

The battery terminal voltage Vbat is measured indirectly by measuring the voltages VB and V-. As shown in Figure 8, Vb and V- are first sensed and conditioned by the voltage and current sense amplifiers before being fed to the 'C240.

Voltage VB is scaled and level shifted to bring the voltage into the range of ADC by using the VB sense amplifier circuit shown in Figure 11.

Figure 11. VB Sense Amplifier



For the amplifier circuit shown in Figure 11, the input voltage ( $V_{bo}$ ) to ADC channel ADCIN7 is calculated by the equation:

$$V_{bo} = 2.5 - \frac{VB(4530 \parallel 49900)}{200000 + (4530 \parallel 49900)}$$

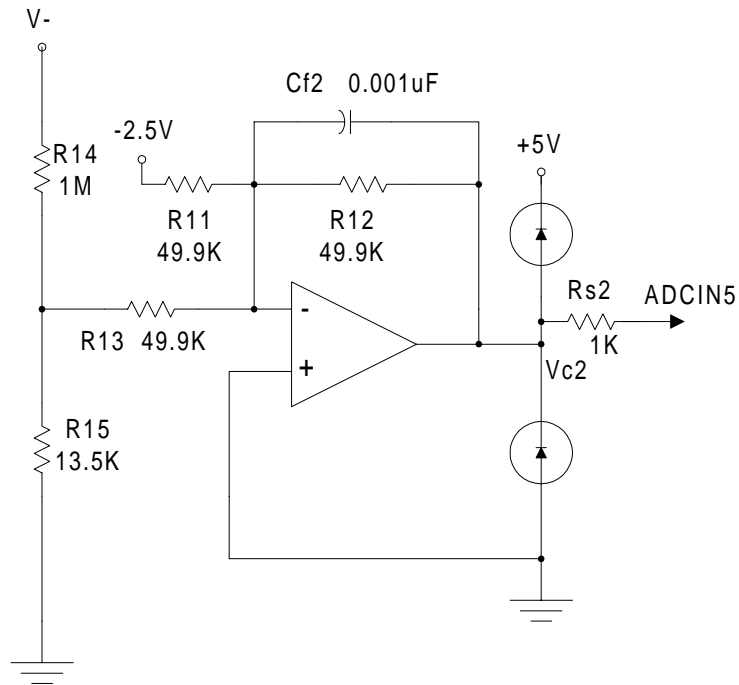
The voltage ( $VB$ ), the corresponding input voltage ( $V_{bo}$ ) to ADC channel ADCIN7, and the resulting ADC data register values are listed in Table 4.

Table 4. Voltage  $VB$  Scaling and Level Shifting

$VB(V)$	$V_{bo}(V)$	ADC_FIFO
+123	0	0000h
0	2.5	7FC0h
-123	5	FFC0h

Voltage  $V-$  is scaled and level shifted to bring the voltage into the range of ADC by using the lower dc bus capacitor voltage sense amplifier circuit shown in Figure 12.

Figure 12. Lower DC Bus Capacitor Voltage (V-) Sense Amplifier



For the amplifier circuit shown in Figure 12, the input voltage (Vc2) to ADC channel ADCIN5 is calculated by the equation:

$$V_{c2} = 2.5 - \frac{(13500 \parallel 49900)V_-}{10^6 + (13500 \parallel 49900)}$$

The voltage (V-), the corresponding input voltage (Vc2) to ADC channel ADCIN5, and the resulting ADC data register values are listed in Table 5.

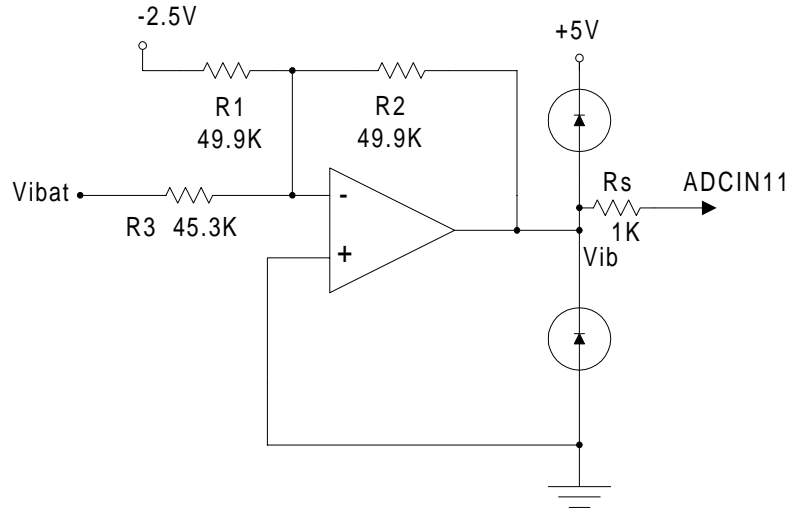
Table 5. Lower DC Bus Capacitor Voltage (V-) Scaling and Level Shifting

V- (V)	Vc2 (V)	ADCFIFO
0	2.5	7FC0h
-238	5	FFC0h



Inductor current ( $I_b$ ) is sensed by the current sensor. The sensor output voltage ( $V_{ibat}$ ) is scaled and level shifted to bring the voltage into the range of ADC by using the battery inductor current sense amplifier circuit shown in Figure 13. The battery inductor current sensor generates 0.16V per ampere of current.

Figure 13. Battery Inductor Current ( $I_b$ ) Sense Amplifier



For the amplifier circuit shown in Figure 13, the input voltage ( $V_{ib}$ ) to ADC channel ADCIN11 is calculated by the equation:

$$V_{ib} = 2.5 - \frac{49900V_{ibat}}{45300}$$

The inductor current ( $I_b$ ), the corresponding sensor output voltage ( $V_{ibat}$ ), the input voltage ( $V_{ib}$ ) to ADC channel ADCIN11, and the resulting ADC data register values are listed in Table 6.

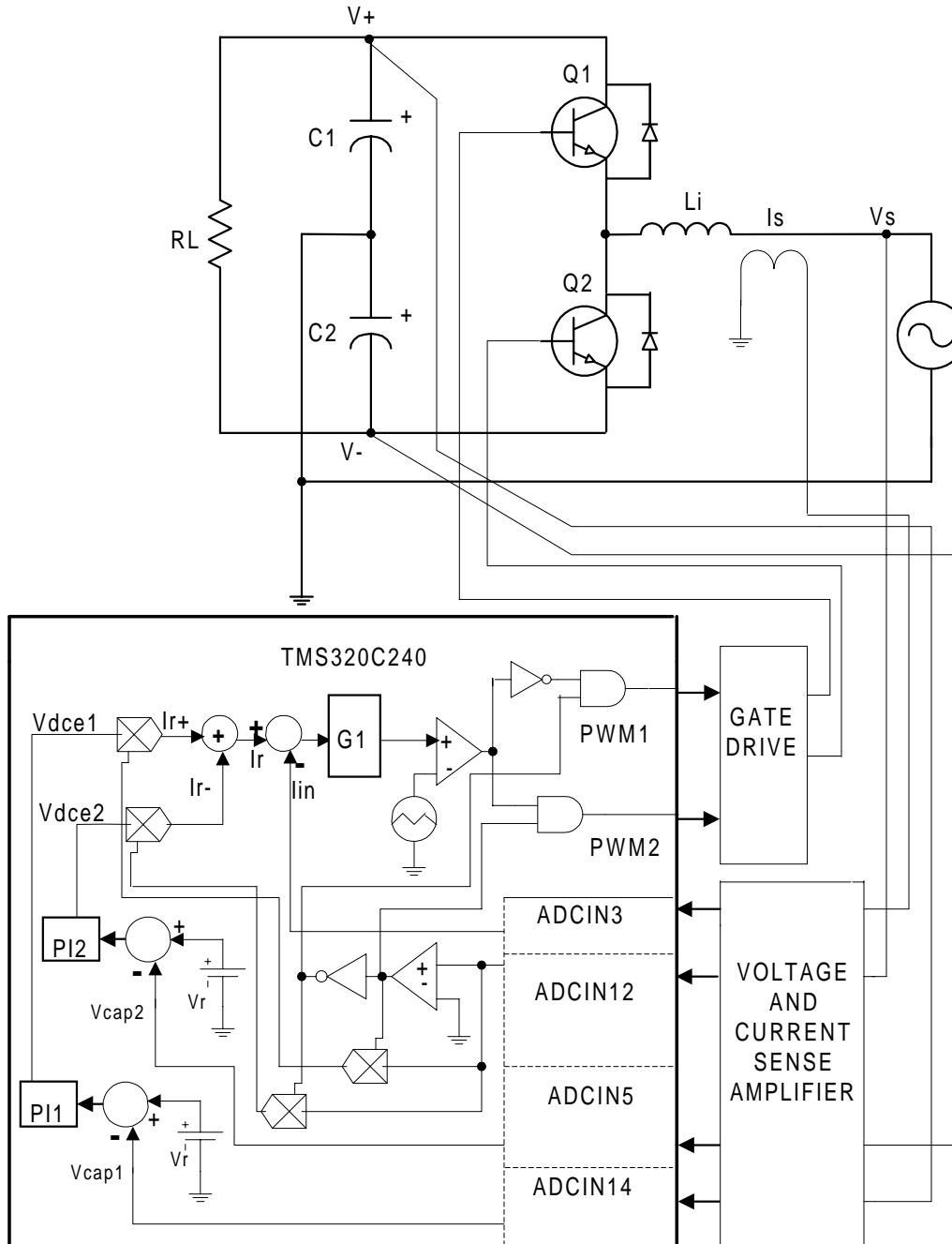
Table 6. Battery Inductor Current Scaling

$I_b$ (A)	$V_{ibat}$ (V)	$V_{ib}$ (V)	ADC_FIFO
+14.2	2.27	0	0000h
0	0	2.5	7FC0h

## Input Power Factor Control (PFC)

Figure 14 shows the input power factor controller (PFC) stage interfaced to the 'C240.

Figure 14. Input PFC Stage Implementation





The PFC stage consists of power devices Q1 and Q2, the dc bus capacitors C1 and C2, and the input inductor  $L_i$ . This is an ac-dc boost converter, which converts the ac input voltage to a high dc bus voltage and maintains sinusoidal input current at high input power factor. As indicated in Figure 14, four signals are required to implement the control algorithm:

- Input voltage,  $V_s$
- Input inductor current,  $I_s$
- Upper DC bus capacitor voltage,  $V_+$
- Lower DC bus capacitor voltage,  $V_-$

The converter is controlled by two feedback loops. The average output dc voltage is regulated by a slow response outer loop; whereas, the inner loop that shapes the input current is a much faster loop.

The system parameters used in this design are:

- $P_{out} = 2000W$
- $V_{dc1} = V_{dc2} = 200V$
- $V_{dc} = 400V$
- $F_s = 20kHz$
- $L = 300\mu H$
- $C_1 = C_2 = 4500\mu F$
- $R_l = 160 \text{ ohms}$

As shown in Figure 14, the instantaneous signals  $V_+$ ,  $V_-$ ,  $V_s$ , and  $I_s$ , are all sensed and conditioned by the voltage and current sense amplifiers inside the block labeled VOLTAGE AND CURRENT SENSE AMPLIFIER. This block is further explained in the *Input PFC Voltage and Current Sensing* section. The sensed signals  $V_+$ ,  $V_-$ ,  $V_s$ , and  $I_s$  are then fed back to the DSP by the four ADC channels ADCIN14, ADCIN5, ADCIN12, and ADCIN3, respectively. The digitized sensed voltages for the upper and lower dc bus capacitors,  $V_{cap1}$  and  $V_{cap2}$ , are each compared to the desired reference  $V_r$ . The difference between the reference  $V_r$  and each of the voltages  $V_{cap1}$  and  $V_{cap2}$ , are fed into the PI regulators PI1 and PI2, respectively. The output of PI1 and PI2 are  $V_{dce1}$  and  $V_{dce2}$ , respectively. These are multiplied by the sinusoidal input voltage waveform to generate the reference current command for the inner current loop.



In Figure 14,  $I_r$  is the reference current command for the inner current loop.  $I_r$  has sinusoidal wave shape and its amplitude is such that it maintains the output dc voltage at a reference level  $V_r$ , against variation in load and fluctuation in line voltage from its nominal value. The positive and negative half cycles of  $I_r$  are  $I_{r+}$  and  $I_{r-}$ , respectively. The amplitude of  $I_{r+}$  is such that the voltage across capacitor C1 is maintained at the reference voltage level  $V_r$  during the positive half cycle of the input supply voltage. Similarly, the amplitude of  $I_{r-}$  is such that the voltage across capacitor C2 is maintained at the reference voltage level  $V_r$  during the negative half cycle of the input supply voltage. The waveform of  $I_{r+}$  is obtained by multiplying the positive half of the input sinusoidal voltage with  $V_{dce1}$ , where  $V_{dce1}$  is the output of the regulator PI1. The waveform of  $I_{r-}$  is obtained by multiplying the negative half of the input sinusoidal voltage with  $V_{dce2}$ , where  $V_{dce2}$  is the output of the regulator PI2.

The PI regulators PI1 and PI2 are both based on  $G_v(s)$ , where,

$$G_v(s) = \frac{1 + 3.405 \times 10^{-3} s}{3.405 \times 10^{-3} s}$$

A current sensor is used to sense the actual input inductor current,  $I_s$ . The sensed digitized inductor current is  $I_{in}$ . The difference between  $I_r$  and  $I_{in}$  is passed into the regulator G1 based on  $G_c(s)$ , where

$$G_c(s) = 2.5 \times 10^6 \times \frac{s + 6.283 \times 10^3}{s^2 + 25.133 \times 10^3 s}$$

The output of this current regulator is used to generate the PWM gating signals, PWM1 and PWM2, with the desired duty cycle. This current regulator output is first converted to a proportional Q0 number and then passed onto the PWM module through the full compare register CMPR1. This CMPR1 value is compared with a 20kHz triangle waveform generated by Timer1 inside the PWM module. This generates the PWM signals PWM1 and PWM2 that drive the switches Q1 and Q2, respectively. During the positive half cycle of the input voltage, Q2 is in PWM mode and Q1 is turned off. This is accomplished by configuring the action control register ACTR for forced low PWM1 and for active high PWM2. During the negative half cycle of the input voltage, Q1 is in PWM mode and Q2 is turned off. Again, this is accomplished by configuring ACTR for forced low PWM2 and for active high PWM1.

## PFC Controller Implementation

The current controller  $G_c(s)$  is transformed to the equivalent digital form, as shown below, before being implemented by the 'C240.

Beginning with the analog controller:

$$G_c(s) = \frac{K_i (s + a)}{s(s + b)}$$

where,

$$K_i = 2.5 \times 10^6, a = 6283, b = 25133$$



Using bilinear transformation

$$s = \frac{2(z-1)}{T(z+1)},$$

and substituting

$$Gc(s) = \frac{U_I(s)}{E_I(s)},$$

the current controller can be expressed as,

$$\frac{U_I(z)}{E_I(z)} = \frac{K_0 + K_1 z^{-1} + K_2 z^{-2}}{1 - K z^{-1} - K_{U2} z^{-2}}$$

From this, the final form of the digital current controller is,

$$U_I(n) = K[U_I(n-1) + k_{U2}U_I(n-2) + k_2E_I(n-2)] + K_1E_I(n-1) + K_0E_I(n)$$

where,

$$K = \frac{4}{(2 + bT)} = 1.2283,$$

$$k_{U2} = \frac{K_{U2}}{K} = \frac{(bT - 2)}{K(bT + 2)} = -0.18585,$$

$$k_2 = \frac{K_2}{K} = \frac{K_i T(aT - 2)}{2(2 + bT)K} = -26.34,$$

$$K_1 = \frac{K_i T(2aT)}{2(2 + bT)} = 12.06$$

$$K_0 = \frac{K_i T(2 + aT)}{2(2 + bT)} = 44.41,$$

$$T = 50 \times 10^{-6} \text{ seconds}$$

The analog voltage controller:

$$Gv(s) = \frac{1 + 3.405 \times 10^{-3} s}{3.405 \times 10^{-3} s} = K_p + \frac{K_I}{s}$$



where,

$$K_p = 1, K_I = 293.7$$

Using bilinear transformation and substituting

$$G_V(s) = \frac{U_V(s)}{E_V(s)},$$

the voltage controller can be expressed as,

$$\frac{U_V(z)}{E_V(z)} = \frac{K_0 + K_1 z^{-1}}{1 - z^{-1}}$$

Therefore, the final form of the digital voltage controller is,

$$U_V(n) = U_V(n-1) + K_0 E(n) + K_1 E(n-1)$$

where,

$$K_0 = K_p + \frac{K_I T}{2} = 1,$$

$$K_1 = -K_p + \frac{K_I T}{2} = -0.985,$$

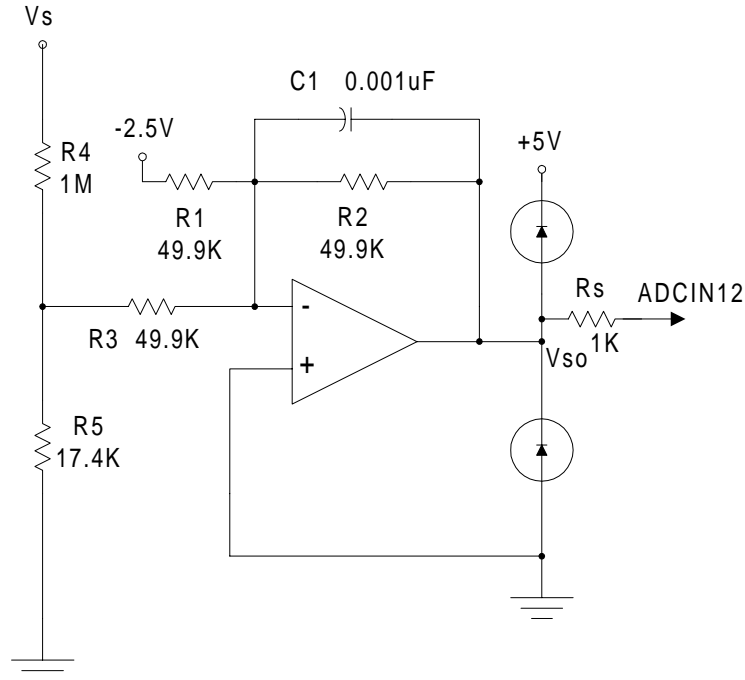
$$T = 100 \times 10^{-6} \text{ seconds}$$

## Input PFC Voltage and Current Sensing

As shown in Figure 14, the instantaneous signals  $V_s$ ,  $I_s$ ,  $V_+$ , and  $V_-$  are all sensed and conditioned by the voltage and current sense amplifiers before being fed to the 'C240.

Input supply voltage ( $V_s$ ) is scaled and level shifted to bring it into the range of ADC by using the UPS input voltage sense amplifier circuit shown in Figure 15.

Figure 15. UPS Input Voltage ( $V_s$ ) Sense Amplifier



For the amplifier circuit in Figure 15, the input voltage ( $V_{so}$ ) to ADC channel ADCIN12 is calculated by the equation:

$$V_{so} = 2.5 - \frac{V_s(17400 \parallel 49900)}{10^6 + (17400 \parallel 49900)}$$

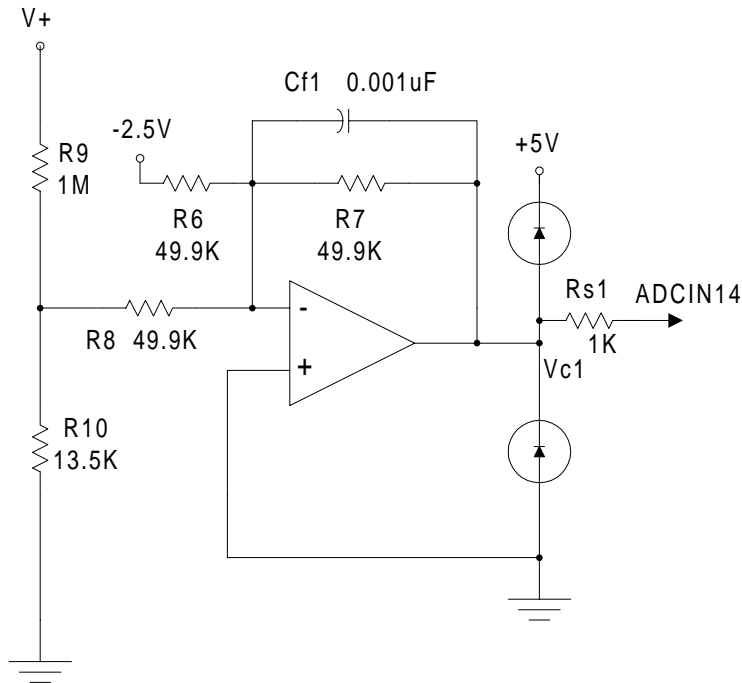
The input voltage ( $V_s$ ), the corresponding input voltage ( $V_{so}$ ) to ADC channel ADCIN12, and the resulting ADC data register values are listed in Table 7.

Table 7. UPS Input Voltage Scaling and Level Shifting

$V_s$ (V)	$V_{so}$ (Vdc)	ADCFIFO
+196	0	0000h
0	2.5	7FC0h
-196	5	FFC0h

The upper DC bus capacitor voltage (V+) is scaled and level shifted to bring it into the range of ADC by using the upper dc bus capacitor voltage sense amplifier circuit shown in Figure 16.

Figure 16. Upper DC Bus Capacitor Voltage (V+) Sense Amplifier



For the amplifier circuit in Figure 16, the input voltage (Vc1) to ADC channel ADCIN14 is calculated by the equation:

$$V_{c1} = 2.5 - \frac{(13500 \parallel 49900)V_+}{10^6 + (13500 \parallel 49900)}$$

The voltage (V+), the corresponding input voltage (Vc1) to ADC channel ADCIN14, and the resulting ADC data register values are listed in Table 8.

Table 8. Upper DC Bus Capacitor Voltage (V+) Scaling

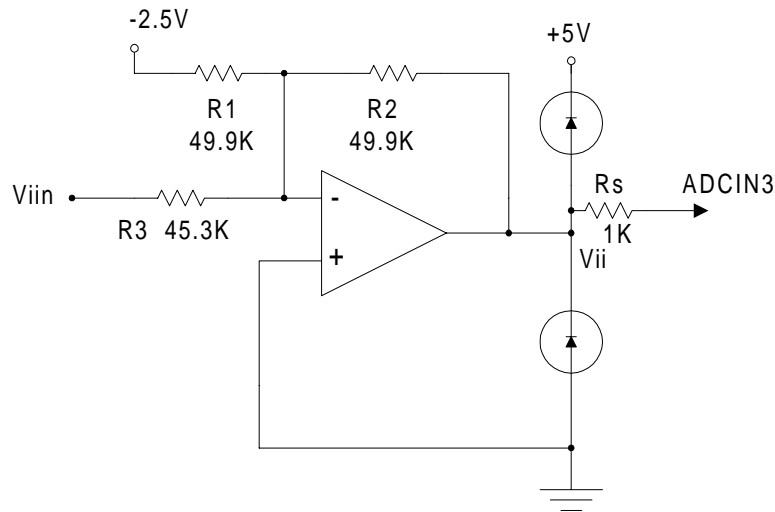
V+ (V)	Vc1 (V)	ADCFIFO
0	2.5	7FC0h
+238	0	0000h



The lower DC bus capacitor voltage (V-) sense amplifier circuit is the same circuit as in the battery charger stage (Figure 12), explained in the *Battery Charger Voltage and Current Sensing* section.

PFC input inductor current ( $I_s$ ) is sensed by the current sensor. The sensor output voltage ( $V_{iin}$ ) is scaled and level shifted to bring the voltage into the range of ADC by using the PFC input inductor current sense amplifier circuit shown in Figure 17. The PFC inductor current sensor generates 0.16V per ampere of current.

Figure 17. PFC Input Inductor Current ( $I_s$ ) Sense Amplifier



For the amplifier circuit shown in Figure 17, the input voltage ( $V_{ii}$ ) to ADC channel ADCIN3 is calculated by the equation:

$$V_{ii} = 2.5 - \frac{49900V_{iin}}{45300}$$

The PFC input inductor current ( $I_s$ ), the corresponding sensor output voltage ( $V_{iin}$ ), the input voltage ( $V_{ii}$ ) to ADC channel ADCIN3, and the resulting ADC data register values are listed in Table 9.

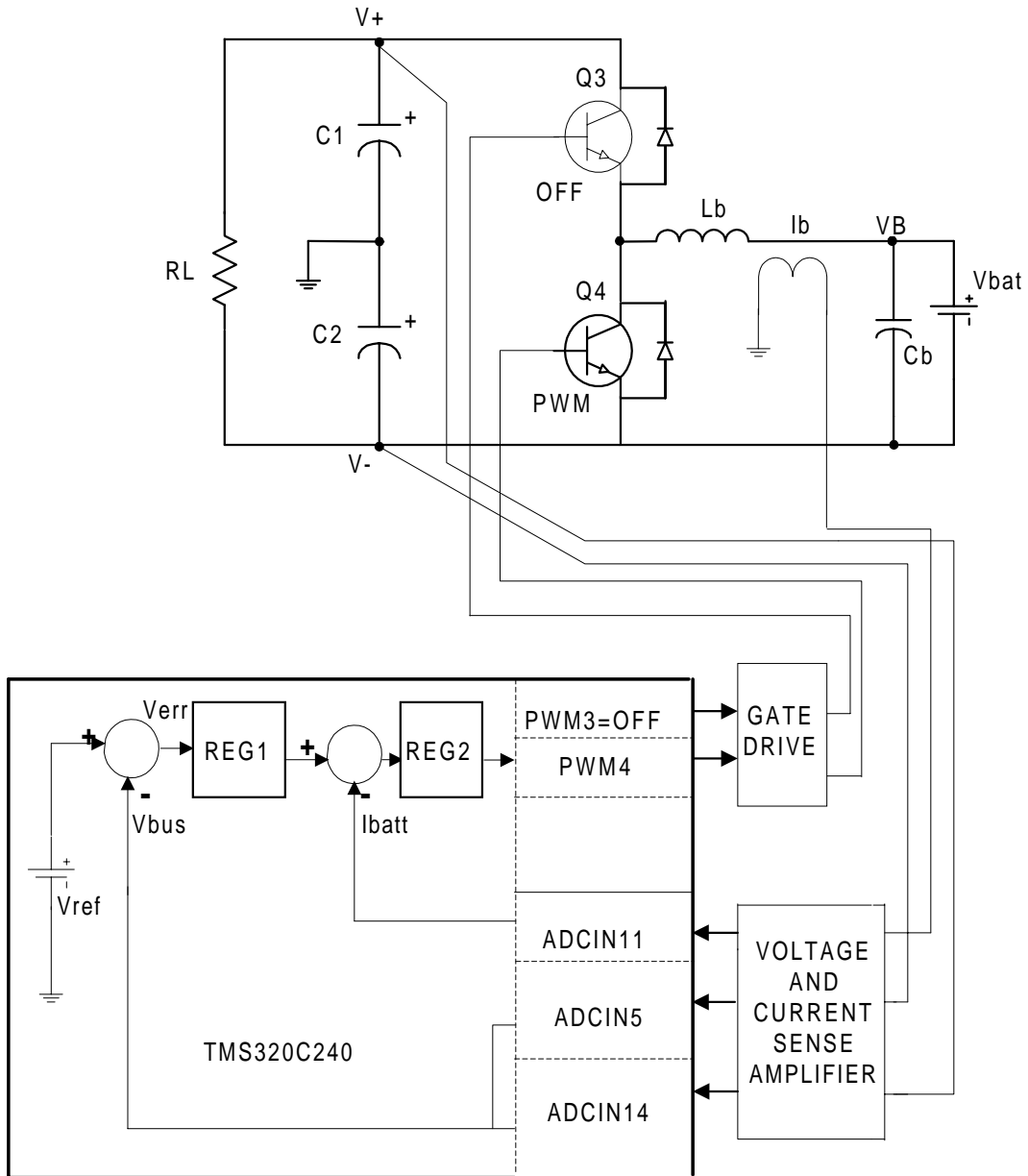
Table 9. PFC Input Inductor Current Scaling and Level Shifting

$I_s$ (A)	$V_{iin}$ (V)	$V_{ii}$ (V)	ADCFIFO
+14.2	2.27	0	0000h
0	0	2.5	7FC0h
-14.2	-2.27	5	FFC0h

## Battery Voltage Boost Stage

Figure 18 shows the battery voltage boost stage interfaced to the 'C240.

Figure 18. UPS Battery Voltage Boost Stage Control





This stage is composed of power device Q4, dc bus capacitors C1 and C2, and boost inductor  $L_b$ . This is basically a dc/dc boost converter, which converts the 110 Vdc battery voltage to 400 Vdc bus. Two signals are required to implement the control algorithm, the battery inductor current  $I_b$  and the dc bus voltage  $V_{bus}$ . The voltage  $V_{bus}$  is measured indirectly by measuring the two bus capacitor voltages,  $V_+$  and  $V_-$ , and then calculating  $V_{bus}$  from these capacitor voltages.  $V_+$  is the capacitor C1 positive terminal voltage with respect to GND and  $V_-$  is the capacitor C2 negative terminal voltage with respect to GND.

As shown in Figure 18, the instantaneous voltages  $V_+$  and  $V_-$ , and the inductor current  $I_b$  are first sensed and conditioned by the respective sense amplifier circuits in the block labeled VOLTAGE AND CURRENT SENSE AMPLIFIER. This block is further explained in the *Boost Stage Voltage and Current Sensing* section. The sensed signals  $V_+$ ,  $V_-$ , and  $I_b$  are then fed back to the DSP by the three ADC channels ADCIN14, ADCIN5, and ADCIN11, respectively.

The bus voltage  $V_{bus}$  is calculated from the two signals  $V_+$  and  $V_-$  as:

$$V_{bus} = (V_+) - (V_-)$$

This calculated voltage,  $V_{bus}$ , is compared to the desired reference bus voltage  $V_{ref}$ . The difference between these two voltages is fed into the voltage controller REG1 based on  $G_v(s)$ , where

$$G_v(s) = 1 + \frac{20}{s}$$

The output of this controller is the reference current command for the inner current loop. This reference current is compared with boost inductor current feedback  $I_{batt}$  and then the difference is passed to the current controller REG2 based on  $G_i(s)$ , where,

$$G_i(s) = \frac{0.2338(s + 1250)}{s + 3120.81}$$

The output of this current controller is the command voltage, which is used to determine the duty cycle of the PWM gating signal. The current regulator output is first converted to a proportional Q0 number and then passed onto the PWM module through the full compare register CMPR2. The PWM module compares this value with a 20kHz triangle waveform generated internally by Timer1. This generates the required PWM signal PWM4, which controls the switch Q4. During the boost operation Q4 is in PWM mode and Q3 is turned off. This is accomplished by configuring the action control register ACTR for active high PWM4 and forced low PWM3.



## Boost Stage Controller Implementation

The current controller is transformed to the equivalent digital form, as shown below, before being implemented by the 'C240.

Beginning with the analog controller:

$$G_I(s) = \frac{K_i(s+a)}{s+b}$$

where,

$$K_i = 0.2338, a = 1250, b = 3120.81$$

Using bilinear transformation

$$s = \frac{2(z-1)}{T(z+1)},$$

and substituting

$$G_I(s) = \frac{U_I(s)}{E_I(s)},$$

the current controller can be expressed as,

$$\frac{U_I(z)}{E_I(z)} = \frac{K_0 + K_1 z^{-1}}{1 - K_2 z^{-1}}$$

From the last equation the final form of the digital current controller is,

$$U_I(n) = K_0 E_I(n) + K_1 E_I(n-1) + K_2 U_I(n-1)$$

where,

$$K_0 = \frac{K_i(2+aT)}{2+bT} = 0.2236,$$

$$K_1 = \frac{K_i(aT-2)}{2+bT} = -0.2101,$$

$$K_2 = \frac{2-bT}{2+bT} = 0.8553,$$

$$T = 50 \times 10^{-6} \text{ seconds}$$



The analog voltage controller:

$$G_V(s) = 1 + \frac{20}{s} = K_p + \frac{K_I}{s}$$

where,  $K_p = 1, K_I = 20$

Using bilinear transformation and substituting

$$G_V(s) = \frac{U_V(s)}{E_V(s)},$$

the voltage controller can be expressed as,

$$\frac{U_V(z)}{E_V(z)} = \frac{K_0 + K_1 z^{-1}}{1 - z^{-1}}$$

Therefore, the final form of the digital voltage controller is,

$$U_V(n) = U_V(n-1) + K_0 E(n) + K_1 E(n-1)$$

where,

$$K_0 = K_p + \frac{K_I T}{2} = 1,$$

$$K_1 = -K_p + \frac{K_I T}{2} = -0.999,$$

$$T = 100 \times 10^{-6} \text{ seconds}$$

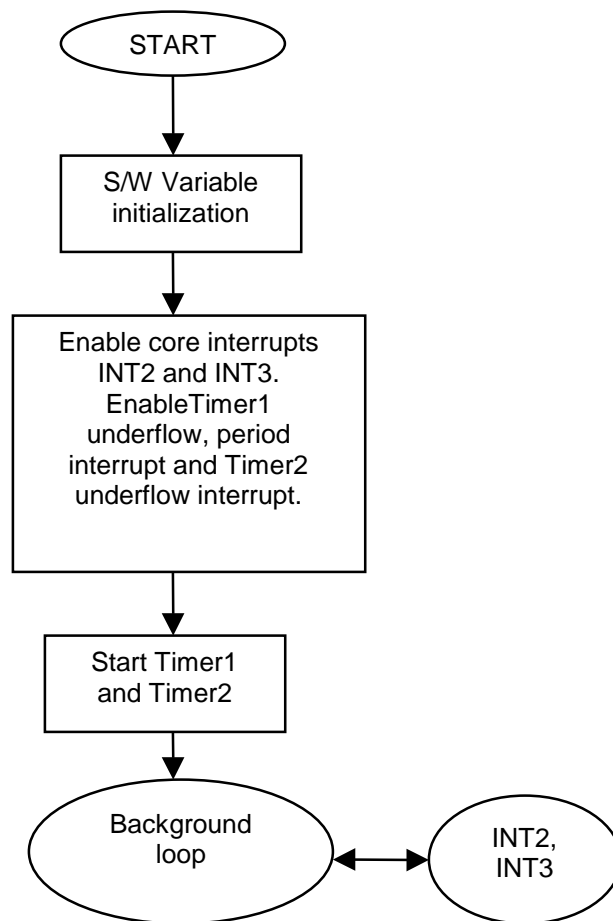
## Boost Stage Voltage and Current Sensing

As shown in Figure 18, the instantaneous signals  $I_b$ ,  $V_-$ , and  $V_+$  are all sensed and conditioned by the voltage and current sense amplifiers before being applied to the 'C240. The  $V_+$  sense amplifier circuit is the same circuit as in the PFC stage (Figure 16), explained in the *Input PFC Voltage and Current Sensing* section. The  $V_-$  and  $I_b$  sense amplifier circuits are the same circuits as in the battery charger stage (Figure 12 and Figure 13) explained in the *Battery Charger Voltage and Current Sensing* section.

## Software Organization

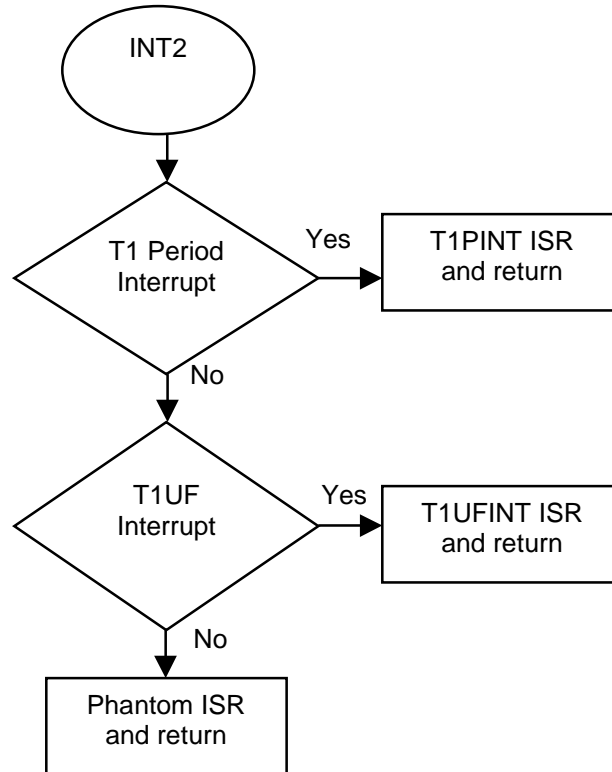
Figure 19 shows the flowchart for the main program. First, the program initializes all the variables. Then it enables the desired interrupts, starts the timers, and loops in the background routine performing all the non time critical functions. CPU interrupts INT2 and INT3 stop execution of this background routine and the program branches to the corresponding interrupt service routines.

Figure 19. Main Program Flowchart



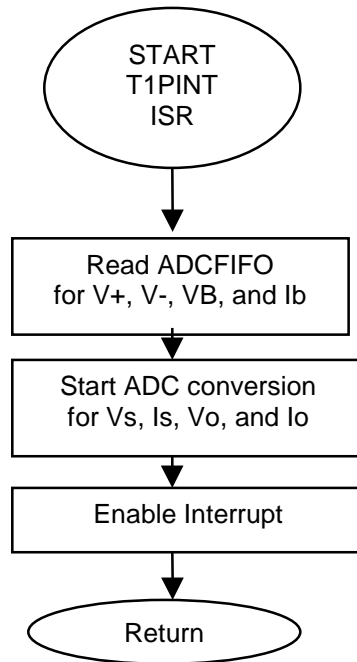
INT2 interrupt sources are Timer1 period and underflow interrupt. As shown in Figure 20, once this is determined, the program branches to the corresponding interrupt service routine.

Figure 20. INT2 Interrupt Dispatcher Flowchart



In the Timer1 period interrupt service routine, the program reads four converted signals from the ADC registers and then starts conversion of four new signals. This is shown Figure 21.

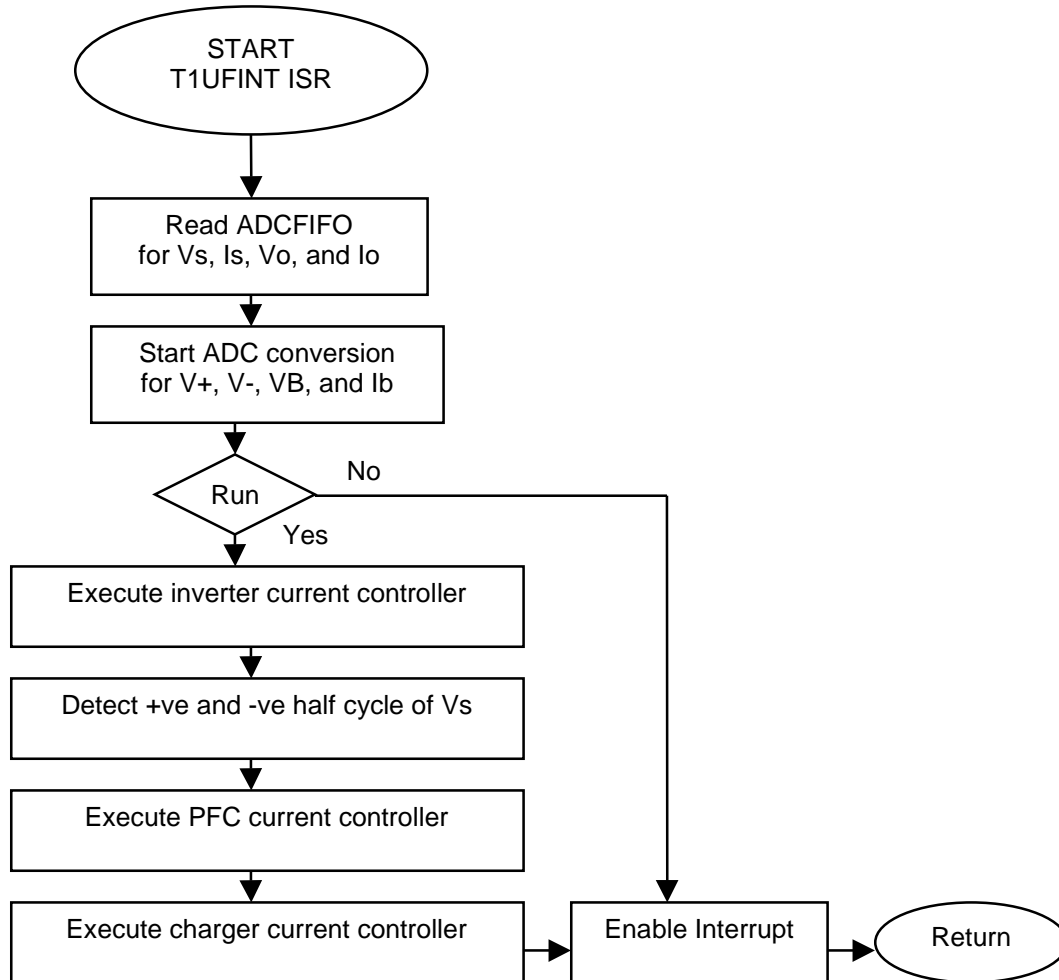
Figure 21. T1PINT Interrupt Service Routine Flowchart





In the Timer1 underflow interrupt service routine, the program reads four converted signals from the ADC registers and then starts conversion of four new signals. Then it executes the required current control algorithms and returns from the interrupt service routine. For normal operation of UPS, this is shown in Figure 22. During back-up mode, the program will not execute the +ve/-ve half cycle detect algorithm, the PFC controller, and the charger controller. Instead, it executes the battery boost current controller and the inverter current controller.

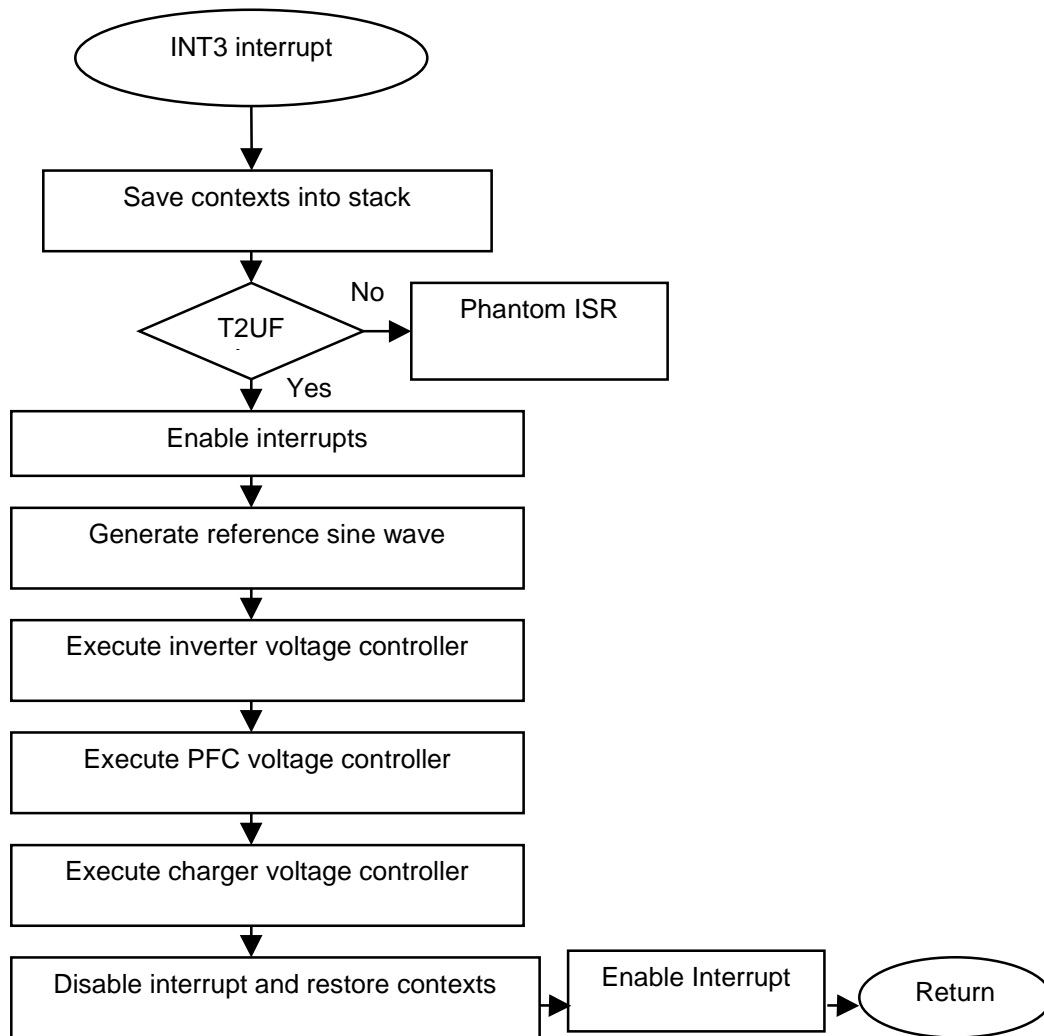
Figure 22. T1UFINT Interrupt Service Routine Flowchart





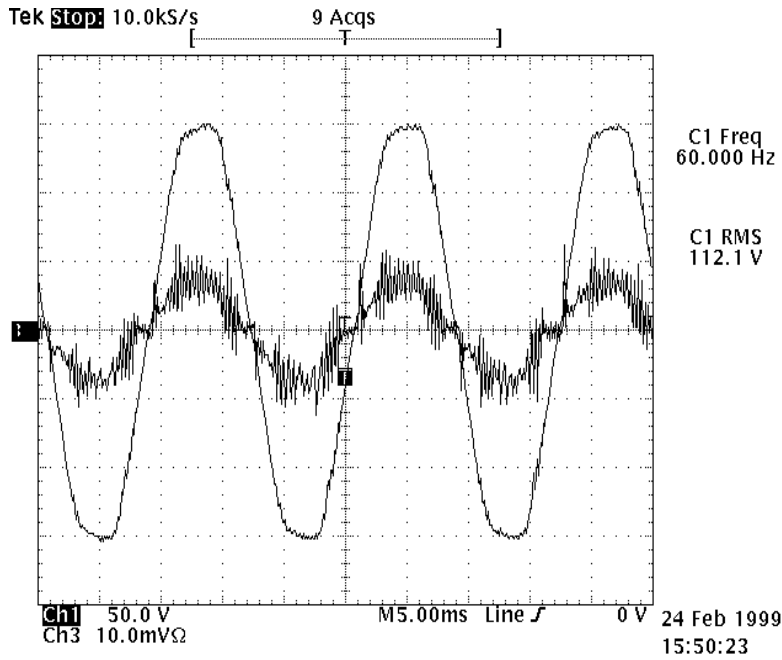
The only source of INT3 interrupt is Timer2 underflow. As shown in Figure 23, once this is determined, the program branches to the T2 underflow interrupt service routine. Prior to branching to the ISR, the contexts are saved in the stack. This is because T2 underflow interrupt is made interruptible by T1 interrupts. In the T2UF interrupt service routine, the program enables the interrupts to allow servicing of T1 interrupts when they are generated. After enabling interrupts, the program generates the reference sine wave and executes required voltage control algorithms. Once these are completed, interrupts are disabled to restore the context from the stack. Following that, interrupts are reenabled and the program returns from the interrupt service routine.

Figure 23. INT3 Interrupt Dispatcher and Timer2 Underflow ISR Flowchart

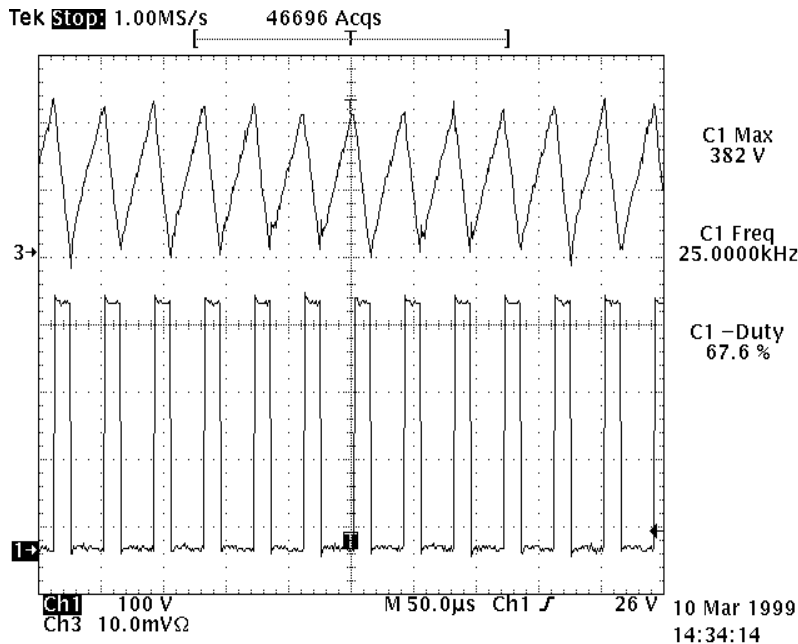




## Experimental Results



UPS Ref Design PFC stage waveforms:  
Ch1 - Input Supply Voltage, Ch2 - Input Current (5A/div)

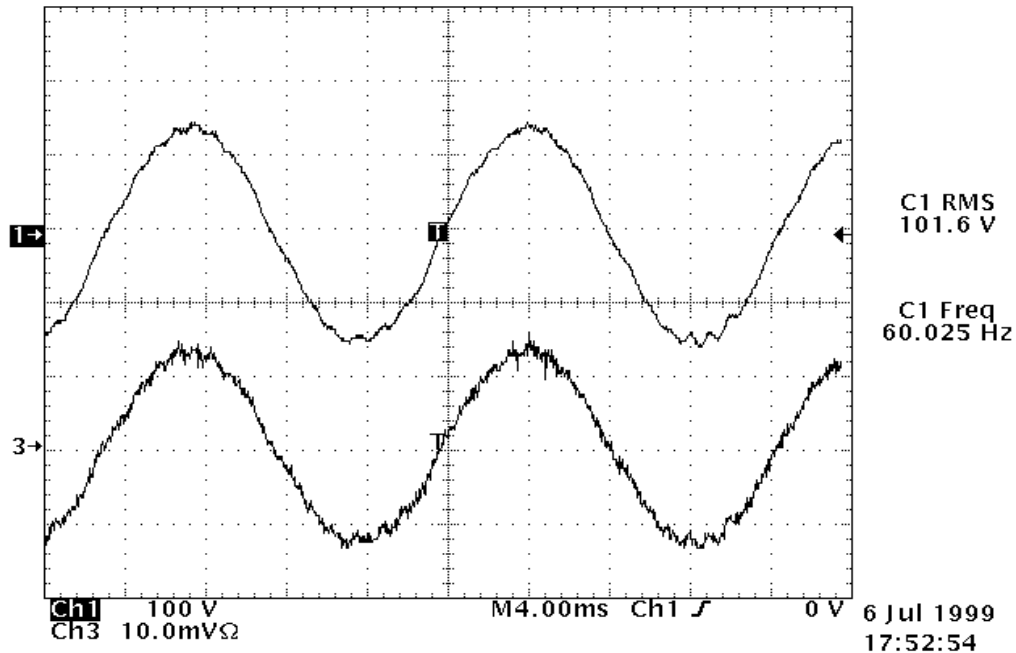


UPS Ref Design :  
Battery Voltage Boost Stage Waveforms:  
Ch1 - Drain to source voltage across Q4, Ch3 - Boost inductor current (2A/div)  
Input - 115VDC, DC bus output voltage - 380V, DC bus load = 250Watt

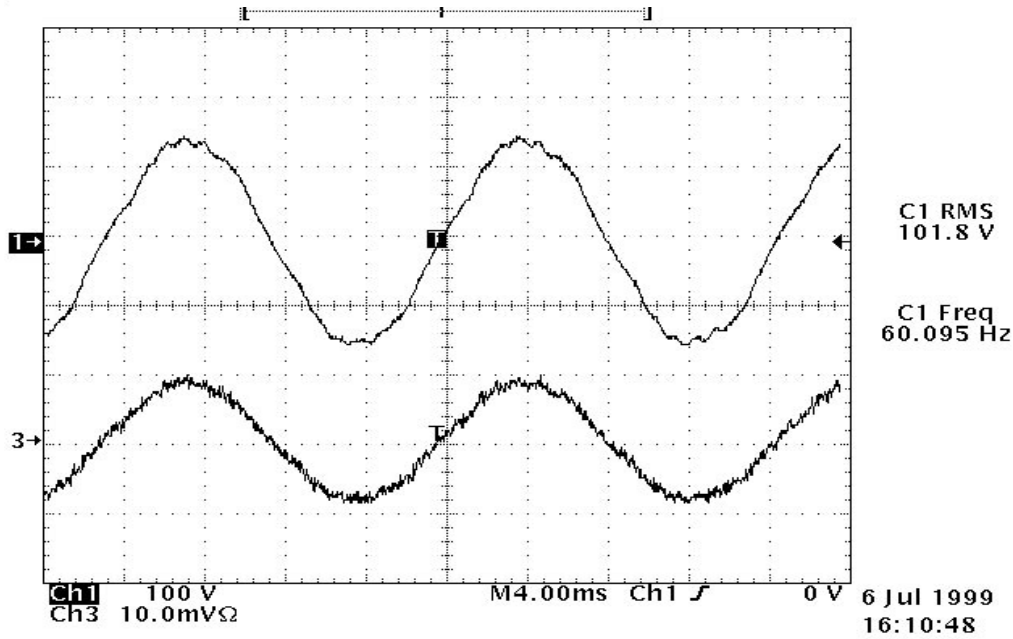


Tek Stop: 25.0kS/s

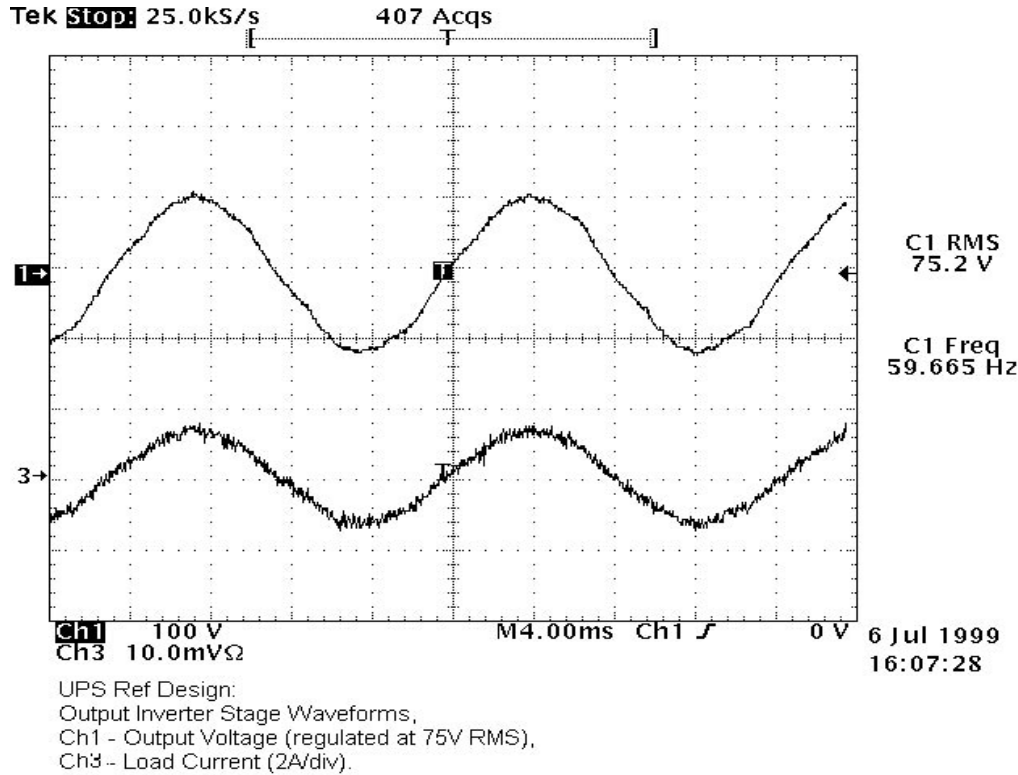
6 Acqs



UPS Ref Design:  
Output Inverter Stage Waveforms,  
Ch1 - Output Voltage (regulated at 102V RMS),  
Ch3 - Load Current (2A/div).



UPS Ref Design:  
Output Inverter Stage Waveforms,  
Ch1 - Output Voltage (regulated at 102V RMS),  
Ch3 - Load Current (2A/div).



## Appendix A Example Code

```

*****
** File Name : UPS.asm          **
** Project   : UPS Reference Design      **
** Originator: Shamim Choudhury        **
**          : Texas Instruments         **
**          : DSP Digital Control Systems Applications      **
** Target   : TMS320C240/F240(EVM)+CPC UPS HARDWARE      **
*****
; Description
;
; This program implements closed loop control of multiple stages of a
; triple conversion on-line UPS system using TMS320C240/F240
;
; For the closed loop control, the program uses eight integrated
; channels of '240 to sense eight signals from the UPS power
; stages. Then it implements all the control algorithms and
; generates six PWM signals to control the power stage switches.
; PWM1 and PWM2 control the Input PFC stage, PWM3 controls the Battery
; Charger stage, PWM4 controls the Battery Boost stage, and, PWM5 and
; PWM6 control the Output Inverter stage.
;

```



```
; The complete program is divided into 16 different modules: 1 main
; module, 5 initialization modules, 4 current control modules, 4
; voltage control modules, 1 zero-crossing and phase determination
; module, and 1 reference sine wave generation module.
;=====
; Debug directives
;-----

.def Vpos
.def Vneg
.def max_Vref
.def Vo
.def VB
.def Vs
      .def V_bus

.def Is
.def Io
.def Ib

;-----
; Peripheral Registers and constants of TMS320C240
;-----
.include "c240.h"
.mmregs

ST0 .set 0 ; status register ST0
ST1 .set 1 ; status register ST1
wd_rst_1 .set 055h ; watchdog timer reset strings
wd_rst_2 .set 0aah ;
LED_addr .set 0Ch ; addr of LED display on EVM
LED_freq .set 3000 ; LED update sub-divider
;=====
```



```
; Variables in B1 page 0
;-----
.bss GPR0,1 ; temporary storage
.bss GPR1,1 ; temporary storage
.bss DAC_HLF_RNG, 1

.bss LED_dir,1 ; LED direction (1: left, 0: right)
.bss LED_data,1 ; LED display

.bss LED_count,1 ; sub-divider counter for LED

.bss Vo,1 ; Output voltage
.bss VB,1 ; Battery voltage (wrt GND)

.bss Vs,1 ; Supply voltage
.bss V_bus,1

.bss Is,1 ; Supply current
.bss Io,1 ; Load current
.bss Ib,1 ; Battery current

.bss max_Vref_trgt,1
;-----
; Context variables
;-----
ST0_save .usect ".context",1; saved ST0 in B2 (DP=0)
ST1_save .usect ".context",1; saved ST1 in B2
ACCH .usect ".extcont",1; acc high in B1 page 0 (DP=6)
ACCL .usect ".extcont",1; acc low in B1 page 0
P_hi .usect ".extcont",1; P high in B1 page 0
P_lo .usect ".extcont",1; P low in B1 page 0
T_save .usect ".extcont",1; T in B1 page 0
AR0_save .usect ".extcont",1; AR0 in B1 page 0
AR1_save .usect ".extcont",1; AR1 in B1 page 0
AR2_save .usect ".extcont",1; AR2 in B1 page 0
stack_ptr .usect ".B0P1",32 ; stack in B0 page 1 (DP=3)

;=====
; Routine Name: main.asm
; Originator : Zhenyu Yu
; Revised by : Shamim Choudhury
; Texas Instruments
; DSP Digital Control Systems Applications
;-----
; Description :
;
; Initializes the variables for proper sampling, control
; loop implementation and PWM generation.
;
; GP Timers 1 and 2 are used as the main time bases to time the
; sampling and control.
;
```



```
; GP Timer 1 is the time base for PWM generation, sampling and
; high-freq control loops. It operates in C-Up/Down mode. It's
; period is 500, giving a frequency of 20KHz for PWM, sampling
; high-freq control loops.
;
; GP Timer 2 is the time base for low-freq control loops. It
; operates in C-UP/Down mode with a period of 1000, giving a
; frequency of 10KHz for low-freq control loops.
;
; AR7 is reserved for use by low-freq interrupt handling
; routine as stack pointer. No other routines are allowed to
; use AR7.
;
; The program allows one level of interrupt nesting. The low-
; freq control loops and routines are interruptible by high-freq
; control loops and routines.
;
; For now, only AR0, AR1 and AR2 are saved as contexts in interrupt
; service routines in addition to ST0/1, ACC, T and P registers.
; More auxiliary registers can be saved if required.
;
;-----
.sect ".vectors"

RESET  BSTART  ; PM 0 Reset Vector
INT1   BPHANTOM; PM 2 Int level 1
INT2   BEVA_ISR; EV interrupt Group A
INT3   BEVB_ISR; EV interrupt Group B
INT4   BPHANTOM; PM 8 Int level 4
INT5   BPHANTOM; PM A Int level 5
INT6   BPHANTOM; PM C Int level 6
RESERVEDBPHANTOM; PM E (Analysis Int)
SW_INT8BPHANTOM; PM 10User S/W int
SW_INT9BPHANTOM; PM 12User S/W int
SW_INT10BPHANTOM; PM 14User S/W int
SW_INT11BPHANTOM; PM 16User S/W int
SW_INT12BPHANTOM; PM 18User S/W int
SW_INT13BPHANTOM; PM 1AUser S/W int
SW_INT14BPHANTOM; PM 1CUser S/W int
SW_INT15BPHANTOM; PM 1EUser S/W int
SW_INT16BPHANTOM; PM 20User S/W int
TRAP   BPHANTOM; PM 22Trap vector
NMI    BPHANTOM; PM 24Non maskable Int
EMU_TRAPBPHANTOM; PM 26Emulator Trap
SW_INT20BPHANTOM; PM 28User S/W int
SW_INT21BPHANTOM; PM 2AUser S/W int
SW_INT22BPHANTOM; PM 2CUser S/W int
SW_INT23BPHANTOM; PM 2EUser S/W int
```





```
.text
START DINT    ; Set global interrupt mask
  SETC OVM    ;Set Overflow Mode
; -----
; Configure system registers
; -----
LDP #0E0h     ; point at Sys Mod reg page 0

SPLK #0100000011000000b,SYSCR ; Make CPUCLK src of
                                ; CLKOUT
SPLK #0000000000100000b,SYSSR ; Clear all SYSSR bits
                                ;(except HP0)

SPLK #01101111b,WD_CNTL  ; Disable the WD timer
SPLK #wd_rst_1,WD_KEY    ; Reset watchdog timer
SPLK #wd_rst_2,WD_KEY

SPLK #10110001b,CKCR1 ; CPUCLK=20MHz if CLKIN=10MHz
; SPLK #10111011b,CKCR1 ; CPUCLK=20MHz if CLKIN=10MHz
; SPLK #11001100b,CKCR1 ; CPUCLK=20MHz if CLKIN=8MHz
; SPLK #11100100b,CKCR1 ; CPUCLK=20MHz if CLKIN=4MHz

SPLK #00000001b,CKCR0 ; Disable and re-enable to activate
                        ; change
PLL_test SPLK #11000001b,CKCR0 ; Wait until PLL is re-enabled.
  BIT CKCR0,BIT5 ; Bits 5,4 are 1x when PLL is
                  ; working
  BCND PLL_test,NTC ; Branch to PLL_test if PLL is not
                    ; locked

;The DAC module requires that wait states be generated for proper
;operation.

LDP #6 ; Point to memory page 0 of B1
SPLK #04,GPR0 ; Set wait state generator
; Program Space, 0 wait states
; Data Space, 0 wait states
; I/O Space, 1 wait state
OUT GPR0,0ffffh ; WSGR <= (GPR0)

; -----
; Initialize ADC module
; -----
LDP #0E0h ; Point at Sys Module reg page 0
SPLK #000000000011b,ADC_CNTL2 ; Disable EV and Ext
;SOC and set p/s
lacl ADCFIFO2 ; Clear ADC result FIFOs
lacl ADCFIFO2
lacl ADCFIFO1
lacl ADCFIFO1
```



```

;-----
; Configure GP Timers
;-----
; Timer 1 period
;  $T_{pwm}/50nS/2 = (50uS/50nS)/2 = 500$ 
T1_period_ .set 500 ; 20KHz for PWM and high-freq sample/control

; Scaled Timer 1 period
T1_periods_ .set 500*32 ; Q5

T2_period_ .set 1000 ; 10KHz for low-freq sample/control
;T3_period_ .set 07FFFh ; Reserved time base

LDP #232 ; Point at EV register page
SPLK #T1_period_,T1CMP ; Zero the duty cycles of T
; cmps
SPLK #T2_period_+1,T2CMP;
; SPLK #T3_period_+1,T3CMP;
SPLK #T1_period_,T1PER ; Set GPT1 pr based on PWM
; freq
SPLK #T2_period_,T2PER ; Set GPT2 pr based on outer
; loop freq
; SPLK #T3_period_,T3PER ; Set period

SPLK #0,T1CNT ; Zero GPT1 counter
SPLK #0,T2CNT ; Zero GPT2 counter
SPLK #0,T3CNT ; Zero GPT3 counter
SPLK #000001010101b,GPTCON ; Set all T cmps active low
SPLK #1010100000000010b,T1CON ; GPT1 in up/dn mode
SPLK #1010100010000010b,T2CON ; GPT2 in up/dn mode in
; synch w GPT1
; SPLK #1001010110000010b,T3CON ; GPT3 in c-up mode
; p/s=32 in synch w T1
;-----
; Configure PWM outputs
;-----
SPLK #00,CMPR1 ; Zero the PWM duty cycles
SPLK #T1_period_,CMPR2 ; Zero the PWM duty cycles.
SPLK #T1_period_,CMPR3 ;

; SPLK #1080h,DBTCON; Define dead band (16*50=0.8uS)
; Enable DB for PWM5,6 only
; SPLK #2080h,DBTCON; Define dead band (32*50=1.6uS)
; Enable DB for PWM5,6 only
; SPLK #40c0h,DBTCON; Define dead band (64*50nS=3.2uS)
; Enable DB for PWM3,4,5,6 only
SPLK #28c0h,DBTCON; Define dead band (40*50nS=2.0uS)
; Enable DB for PWM3,4,5,6 only

; AH = Active High, AL = Active Low, FL = Forced Low
SPLK #0000011010000000b,ACTR ; PWM polarity: 123=FL, 6=AL,
; 45=AH
; For boost PWM3=FL, PWM4=AH
SPLK #0h,IMRA ; Mask PDPINT to enable PWM if safe

```



```

SPLK #0000101110000111b,COMCON
;
;          ||||||||||||||||
;          FEDCBA9876543210
SPLK #1000101110000111b,COMCON; Enable compare/PWM
; operation & outputs
; -----
; Initialize all the modules
; -----
CALL init_xingpll
CALL init_batboost
    CALL init_batcharge
    CALL init_pfc
    CALL init_invrtr
; .
; .
; -----
; Initialize stack pointer
; -----
LAR AR7,#stack_ptr
; -----
; Initialize LED display on EVM
; -----
LDP #6      ; Point to B1 page 0
splk #01h,LED_data ; Set LED display on EVM
out LED_data,LED_addr ; Set LED display
splk #LED_freq,LED_count; reset sub-divider counter
splk #1,LED_dir      ; set LED display direction
; -----
; Initialize variables
; -----
SPLK #0,RUN
SPLK #0,rmp_dly_cnt ; Reset error counter
splk #06000h,max_Vref_trgt ; Max ref volt for each bus
; capacitor
splk #0000h,max_Vref
SPLK #06616h,V_ref ;V_ref=190V
SPLK #1,one ; +1 => one
SPLK #T_sample_,T_sample; sampling period
SPLK #T1_periods_,T1_periods ; max compare value
SPLK #250, HALF_PERIOD
SPLK #1900, DAC_HLF_RNG
SPLK #0,set_F ; zero set F.
SPLK #F_W_,F_W ; Q6, set F to angular speed ratio
SPLK #S_U_,S_U ; Q14, mag of ref voltage
SPLK #min_W_,min_W; Q5, lower limit on set W

SPLK #0,THETAL ; theta low byte
SPLK #0,THETAH ; theta high byte

LAR AR0,#theta_60; point to 1st destination
LAR AR1,#(8-1) ; 8 entries
LACC #angles_ ; point to 1st data item
LARP AR0 ;

```



```
Init_tbTBLR *+,1    ; move and point to next
                    ; destination
    ADD one    ; point to next data item
    BANZ Init_tb,0 ;
    SPLK #theta_i_,theta_i ; Q7, init theta-index ratio
    SPLK #SIN_TABLE_,SIN_1st; init 1st and last entries
                    ; of sin table
    SPLK #(SIN_TABLE_+360),SIN_last

    SPLK #0,THETAH ; zero angular position high
    SPLK #0,THETAL ; zero angular position low

; -----
; Mask/unmask interrupts
; -----
    LDP #232    ; point at EV reg page
    SPLK #0fffh,IFRA ; Clear all Group A int flags
    SPLK #0fffh,IFRB ; Clear all Group B int flags
    SPLK #0fh,IFRC ; Clear all Group C int flags
    SPLK #0281h,IMRA ; Unmask PDPINT,GPT1 UF&PR ints
    SPLK #04h,IMRB ; Unmask GPT2 UF ints
    SPLK #0h,IMRC ; Mask all EV Grp C ints

    LDP #0    ; Point at MMR page
    SPLK #0fffh,IFR ; Clear pending int to CPU
    SPLK #0011110b,IMR; Enable int to CPU (no emu int)

; -----
; Enable GPTs and global interrupt to start real-time operation
; -----
    LDP #232    ; Point at EV reg page
    SPLK #1010100001000010b,T1CON ; Enable the GPTs
    EINT                ; Enable global interrupt

; =====
; Main background loop starts here
; -----
MAIN
    ldp #supply_cycle; set DP for supply cycle flag
        LACC supply_cycle
        BCND pos_cycle, NEQ
    CLRC XF
        B neg_cycle
pos_cycle:
    SETC XF
neg_cycle:

    MAR *,AR0 ; Use AR0
```



```
; -----  
; Call background routines  
; -----  
; CALL Safety_check ; Safety check  
; CALL Clock      ; Global clock routine  
; CALL Parameter  ; Parameter calculation routine  
; CALL Display    ; Display update  
; CALL Comm       ; Communication  
; .  
; .  
; -----  
; Update LED display on EVM  
; -----  
  
lacc LED_count ; load sub_divide counter  
sub one      ; update sub_divide counter  
sac1 LED_count ; time to update LED display?  
BNZ LED_nc   ; no  
splk #LED_freq,LED_count; yes, reset subdivide counter  
bit LED_dir,BIT0 ; left shift?  
bcnd right_shift,NTC ; no  
lacc LED_data,1 ; yes  
sac1 LED_data ; left shift one bit  
bit LED_data,BIT7; time to change direction?  
bcnd LED_update,NTC ; no  
splk #0,LED_dir ; yes  
bLED_update ;  
right_shift lacc LED_data,15 ;  
sach LED_data ; right shift one bit  
bit LED_data,BIT0 ; time to change direction?  
bcnd LED_update,NTC ; no  
splk #1,LED_dir ; yes  
LED_update out LED_data,LED_addr ; update LED display  
LED_nc      ; no update  
  
;Set the desired output frequency  
SPLK #debug_data,set_F  
  
;-----  
; Calculate set angular speed based set F  
;-----  
tag1: LTset_F ; set F -> T: Q15  
MPY F_W ; Q15*Q6=Q21  
PAC ;  
SACH S_W ; -> set angular speed: Q5  
  
SUBH min_W ; Q5,compare W with its upper limit  
BGZ W_in_limit ; continue if within limit  
LACC min_W ; saturate if not  
SACL S_W ;  
W_in_limit
```



```

; -----
; Reset wd timer and loop back
; -----
LDP #0E0h ; point to Sys Mod reg page 0
SPLK #wd_rst_1,WD_KEY ; Reset WD timer
SPLK #wd_rst_2,WD_KEY
BMAIN ; Branch back

; =====
; Level 2 (EV Group A) interrupt dispatcher
; The possible sources are GPT1 UF & PR.
; -----
EVA_ISRST #ST0,ST0_save ; save ST0
SST #ST1,ST1_save ; save ST1

LDP #6 ; point to page 0 of B1
MAR *,AR0 ; use AR0
SACH ACCH ; Save ACC_hi
SACL ACCL ; save ACC_lo
sph P_hi ; Save P_hi
spl P_lo ; save P_lo
mpyk #1 ; P<=T
spl T_save ; save T

SAR AR0,AR0_save ; save AR0
SAR AR1,AR1_save ; save AR1
SAR AR2,AR2_save ; save AR2

LAR AR0,#IVRA ; Read int vetor ID
LACC * ;
SACL GPR0 ; Save to scratch
SUB #027h ; See if the source is GPT1 PR
BCND GPT1_PR,EQ ;
LACC GPR0 ;
SUB #029h ; See if the source is GPT1 UF
BCND GPT1_UF, EQ ;
BPHANTOMA ; got a phantom int

; -----
; GPT1 PR interrupt service
; -----
; GPT1 PR just kicks off sampling of Is, Io, Vs and
; Vo. This is in the middle of the high-freq control loops.
; Since the period of high-freq control loops is 50uS, by the time
; the next period starts, both conversions are done. However, before
; the new sampling is started, old sampled data must be read out.
*****
GPT1_PR

BLDD #ADCFIFO1,Vneg
BLDD #ADCFIFO2,Vpos
BLDD #ADCFIFO1,VB
BLDD #ADCFIFO2,Ib

```



```

LAR AR0,#ADC_CNTL1 ; Sample Vo and Io
SPLK #1111100110101100b,* ;
;   ||||||||||||||||
;
;           5432109876543210
; bit 13 1 Immediate start - 0 = no action over-ride bit 0
; bit 6-4 010 Select channel 10 for ADC2; ADCIN10 samples Io
; bit 3-1 110 Select channel 6 for ADC1; ADCIN06 samples Vo
; bit 0 0 Start of conversion

ADC_test1 BIT *,BIT7 ; make sure ADC is in progress
BCND ADC_test1,NTC

SPLK #1101100111000111b,* ; sample Is and Vs
;   ||||||||||||||||
;
;           5432109876543210
; bit 13 0 no action - 1=Immediate start
; bit 6-4 100 Select channel 12 for ADC2; ADCIN12 samples Vs
; bit 3-1 011 Select channel 3 for ADC1; ADCIN03 samples Is
; bit 0 1 Start conversion after current conversion finishes

BEV_A_end ; Go to the end

; -----
; GPT1 UF interrupt service routine
; -----
; GPT1 UF kicks off sampling of Ib, VB, Vpos and Vneg. Then it executes
; all the high-freq control loops. Sampling
; of these variables completes before the GPT1 PR which is in the
; middle of the high-freq control loops. However, before the new
; sampling is started, old sampled data must be read out.
*****
GPT1_UF
LDP #6
BLDD #ADCFIFO1,Vo ; Read out old sampled data
BLDD #ADCFIFO2,Io ; Make sure the order in which
BLDD #ADCFIFO1,Is ; the data are read here is the
BLDD #ADCFIFO2,Vs ; same as the order they were
; sampled during the last two
; conversions!!!!!!!

LAR AR0,#ADC_CNTL1 ; Sample Vpos and Vneg
SPLK #1111100111101010b,* ;
;   ||||||||||||||||
;
;           5432109876543210
; bit 13 1 Immediate start - 0 = no action over-ride bit 0
; bit 6-4 110 Select channel 14 for ADC2; ADCIN14 samples Vpos
; bit 3-1 101 Select channel 5 for ADC1; ADCIN05 samples Vneg
; bit 0 0 Start of conversion

```



```

ADC_test2 BIT *,BIT7 ; make sure ADC is in progress
    BCND ADC_test2,NTC

    SPLK #1101100110111111b,* ; sample Ib and VB
;   ||||||||||||||||
;   5432109876543210
; bit 13 0no action - 1=Immediate start
; bit 6-4 011 Select channel 11 for ADC2; ADCIN11 samples Ib
; bit 3-1 111 Select channel 7 for ADC1; ADCIN07 samples VB
; bit 01 Start conversion after current conversion finishes

; -----
; Execute high-freq control loops
; -----

    LACC RUN
    BCND bypass_ccnt1,EQ

    CALL Invert_i ; Output inverter current control
    CALL Z_xing_pll ; Detect +ve and -ve half cycle of Vs
    CALL PFCBoost_i ; Input PFC current control
    CALL BatCharge_i ; Battery charger current control
; CALL BatBoost_i ; Battery voltage boost current control
; .
; .

    BEV_A_end

bypass_ccnt1    LDP #232
    SPLK #T1_period_,CMPR2
                SPLK #T1_period_,CMPR3
                SPLK #0000011010000000b,ACTR ;123=FL, 6=AL, 45=AH
    ; For boost configure PWM3=FL, PWM4=AH
    LDP    #dly_cnt_vbb
    SPLK #0,dly_cnt_vbb
    splk #0000h,max_unvbbp
        splk #0000h,max_unvbbm

EV_A_end LDP #6
    LAR AR2, AR2_save ; Restore AR2
    LAR AR1, AR1_save ; Restore AR1
    LAR AR0, AR0_save ; Restore AR0
    lt P_lo    ; T<=P_lo
    mpy #1    ; Restore P_lo
    lph P_hi    ; Restore P_hi
    lt T_save    ; restore T
    LACL ACCL    ; Restore ACC_lo
    ADDH ACCH    ; restore ACC_hi
    LDP #0    ; point to B2
    LST #ST1,ST1_save ; restore ST1
    LST #ST0,ST0_save ; restore ST0
    EINT                ; re-enable int
    RET                ; return

```





```

; =====
; Level 3 (EV Group B) interrupt dispatcher
; The only possible source is GPT2 UF which executes low-freq voltage
; controllers. Contexts are saved to the stack so that the low-
; freq control loops can be made interruptible by high-freq control
; loops.
; -----
EVB_ISR_SST #ST0,ST0_save; save ST0
    SST #ST1,ST1_save; save ST1

    MAR *,AR7    ; Point at stack pointer
    BLDD #ST0_save,++; Save ST0_save
    BLDD #ST1_save,++; Save ST1_save
    SACH *+    ; Save ACC_hi
    SACL *+    ; save ACC_lo
    sph *+    ; Save P_hi
    spl *+    ; save P_lo
    mpyk #1    ; P<=T
    spl *+    ; Save T
    SAR AR0,*+ ; Save AR0
    SAR AR1,*  ; Save AR1

    LAR AR0,#IVRB ; Identify int source
    MAR *,AR0
    LACC *    ;
    SUB #02Dh ; GPT2 UF int?
    BCND GPT2_UF,EQ ; Yes
    BPHANTOMB ; got a phantom int if not

; -----
; Mask emu int before re-enable global int
; -----
GPT2_UF_LAR AR0,#IMR ; Mask emulation int
    LACL *    ;
    AND #01111111b ;
    SACL *    ;
    EINT                                ; Re-enable int

; -----
; Execute low-freq control loops
; -----
    LDP #RUN
        LACC RUN
    BCND bypass_vcntl,EQ

        CALL sinwave_gen ; Generate ref sine wave for inverter
    CALL Invert_v ; Output inverter voltage control
    CALL PFCBoost_v ; Input PFC voltage control
    CALL BatCharge_v ; Battery charger voltage control
; CALL BatBoost_v ; Battery boost voltage control
; .
; .
        BSKIP_BYPASS

```



```

bypass_vcntl:
    ldp #max_Vref
    splk #0000h,max_Vref

SKIP_BYPASS
; -----
; Disable global int to unmask emu int and restore contexts
; -----
    DINT      ; Disable interrupt
    LAR AR0,#IMR ; Unmask emulation int
    LACL *    ;
    OR #1000000b ;
    SACL *    ;

EV_B_end MAR *,AR7
    LAR AR1,*- ; Restore AR1
    LAR AR0,*- ; Restore AR0
    MAR *-    ; Skip T and point at P_lo
    LT *+    ; T<=P_lo. Point back at T
    MPY #1    ; Restore P_lo
    LT *-    ; Restore T. Point at P_lo again
    MAR *-    ; Skip P_lo this time and point at P_hi
    LPH *-    ; Restore P_hi
    LACL *-   ; Restore ACC_lo
    ADDH *-   ; Restore ACC_hi
    LDP #0    ; Point at B2
    BLDD *-,#ST1_save ; Restore ST1_save
    BLDD * ,#ST0_save ; Restore ST0_save

    LST #ST1,ST1_save ; Restore status register ST1
    LST #ST0,ST0_save ; Restore status register ST0
    EINT                      ; Re-enable int
    RET                        ; Return
;=====
; Routine Name: init_invrtr.asm
; Originator   : Shamim Choudhury
;               Texas Instruments
;               DSP Digital Control Systems Applications
;-----
;Description:
;
;Initializes inverter stage variables and coefficients
;-----
init_invrtr:
    ldp #K0_vinv ; set DP for inverter section
        SPLK #02000H,Kmv ;Q22
    SPLK #03000H,Kmi ;q22
    SPLK #00000H,Uqs ;Q15

; PID variables and PID coefficients for voltage control loop
        SPLK #07fffH,K0_vinv ;Q15(K0=1)
        SPLK #00507H,K1_vinv ;Q12(K1=0.31416)
        SPLK #00507H,Kcorr_vinv ;Q12(K1/K0=0.31416)

```



```
        SPLK #00000H,En0_vinv
        SPLK #00000H,GPR0_vinv ;Q15
        SPLK #00000H,Unvinv_H_0 ;Q15
        SPLK #00000H,Un_vinv

; PID variables and PID coefficients for current control loop

        SPLK #00400H,K0_iinv ;Q9 format (k0=2)
            SPLK #001E3H,K1_iinv ;Q9 format (K1=0.9425)
        SPLK #00F14H,Kcorr_iinv ;Q13(K1/K0=0.4712)

            SPLK #00000H,En0_iinv ;Q13
            SPLK #00000H,Uniinv_H_0 ;Q11
            SPLK #00000H,Un_iinv

        ret

;=====
; Routine Name: Invert_v.asm
; Originator : Shamim Choudhury
; Texas Instruments
; DSP Digital Control Systems Applications
;-----
; Description:
;
; Implements inverter stage voltage control algorithm
;
;-----
; Debug directives
        .def K0_vinv
        .def K1_vinv
        .def En0_vinv
        .def GPR0_vinv
        .def Unvinv_H_0

;-----
; Variables in B1 page 0
;-----
        .bss K0_vinv,1
        .bss K1_vinv,1
        .bss En0_vinv,1
        .bss GPR0_vinv,1
        .bss Unvinv_H_0,1
        .bss Vout,1
            .bss Uqs,1
        .bss Kcorr_vinv,1
        .bss Un_vinv,1
        .bss epi_v_o,1
        .bss Upi_v_o,1
        .bss Kmi,1
```



```

;-----
Invert_v:
    SETC SXM

    LDP #K0_vinv

    spm 1
    LACC Vo,10 ;
    sach GPR0_vinv
    lacc GPR0_vinv
    and #03ffh
        sub #512
    neg
    sub #1
    sac1 GPR0_vinv ;Q0
    lt GPR0_vinv
    mpy Kmv ;Q22
    pac ;Q23
    rpt #7
    norm *
    sach Vout ;Q15

    SPM 0

    lacc Uq ;Q15
    sub Vout ;Q15
    SACL En0_vinv ;Store error(Q15)

    lacc Un_vinv,15 ;ACC = Un_vinv(Q30,32-bit)

    LT En0_vinv ;
    MPY K0_vinv ;P<- K0*En0,Q15*Q15

    APAC ;ACC <-- Un_vinv + K0*En-0, Q30
        norm * ;Q31
    sach Upi_v_o ;Q15
    SPM 0

        LACC Upi_v_o
        ADD #07ff0H
        BCND SAT_MINUS,LT

        LACC Upi_v_o
        SUB #07ff0H
        BCND SAT_PLUS, GEQ
        lacc Upi_v_o
    sac1 Unvinv_H_0 ;Q15
    B FWD1

SAT_MINUS
    SPLK #08010h,Unvinv_H_0
    B FWD1

```



```

SAT_PLUS
    SPLK    #07ff0h,Unvinv_H_0
FWD1:
    LACC Unvinv_H_0    ;Q15
    SUB Upi_v_o        ;Q15
    sacl epi_v_o        ;Q15

    lt epi_v_o          ;Q15
    mpy Kcorr_vinv     ;Q12
    pac                ;Q27

    lt En0_vinv        ;Q15
    mpy K1_vinv        ;P<- K1*En1, Q15*q12
    apac              ;Q27

    ADD Un_vinv,12     ;Q27,ACC <-- Un_vinv + K1*En-0 + K0*En-0
                    rpt #3
    norm *            ;q31
    sach Un_vinv        ;Q15

    RET                ; return
;End of routine Invert_v

;=====
; Routine Name: Invert_i.asm
; Originator   : Shamim Choudhury
;               Texas Instruments
;               DSP Digital Control Systems Applications
;-----
; Description:
;
; Implements PI control on current error and generates PWM5 and PWM6.
;
;-----
; Debug directives
;-----
    .def Ta
    .def K0_iinv
    .def K1_iinv
    .def En0_iinv
    .def En1_iinv
    .def Uniinv_H_0

;-----
; Variables in B1 page 0
;-----
    .bss rmp_dly_cnt,1
    .bss GPR2,1
    .bss HALF_PERIOD,1

        .bss Kmv,1 ;

        .bss K0_iinv,1
    .bss K1_iinv,1

```



```

.bss En0_iinv,1
.bss En1_iinv,1

.bss Uniinv_H_0,1
.bss Iout,1

.bss Kcorr_iinv,1
.bss Un_iinv,1
.bss epi_i_o,1
.bss Upi_i_o,1

;-----
Invert_i:
    SETC SXM

    LDP    #K0_iinv

    LACC Io,10 ;
        sach GPR0
    lacc GPR0
    and #03ffh
        sub #512
    neg
        sub #1
    sacl GPR0 ;Q0

        SPM 1
    lt GPR0
    mpy Kmi ;Q22
    pac ;Q23
    rpt #7
    norm * ;q31
    SACH Iout ;Q15

;Subtracts ADC output(current sample) from current command and
;computes error.
;Implements PI control on error and then update compare value to
;generate PWM5 and PWM6.

    SPM 0

    LACC Unvinv_H_0 ;Q15
        SUB Iout ;Q15
    SACL En0_iinv;Q15

    lacc Un_iinv,13 ;ACC(32-bit)(Q24), Un_iinv(Q11)

    LT En0_iinv ;ACC<- Un-1 + K1*En-1, Q24,
    MPY K0_iinv ;P<- K0*En0,Q9*Q15

    APAC ;ACC <-- Un-1 + K1*En-1 + K0*En-0, Q24

```



```

        rpt #2
        norm *      ;Q27
sach Upi_i_o      ;Q11

        LACC Upi_i_o
        ADD #07ff0H
        BCND SAT_MINUS_IO,LT      ;If upi is more -ve than -16
        ; then saturate at max -ve U

        LACC Upi_i_o
SUB   #07ff0H
        BCND SAT_PLUS_IO, GEQ
        lacc Upi_i_o
sac1 Uniinv_H_0 ;Q11
B     FWD_IO

SAT_MINUS_IO
        SPLK #08010h,Uniinv_H_0 ;Q11,neg max = -16
        B     FWD_IO

SAT_PLUS_IO
        SPLK #07ff0h,Uniinv_H_0      ;Q11,pos max = 16
FWD_IO:

LACC Uniinv_H_0 ;Q11
SUB Upi_i_o      ;Q11
sac1 epi_i_o      ;Q11

lt epi_i_o      ;Q11
mpy Kcorr_iinv      ;Q13
pac              ;Q24

lt En0_iinv      ;Q15
mpy K1_iinv      ;P <- K1*En1, Q9*Q15
apac             ;Q24
ADD Un_iinv,13   ;Q24

rpt #2
norm *      ;q27

sach Un_iinv      ;Q11

;Convert Q11 value to an absolute Q0 for use in Compare reg.
spm 3
LT Uniinv_H_0 ; (Q11)
MPY #1000      ;P = 1000*U = 2T*U
PAC           ;ACC = 2T*U/64 = (T/2)*(U/16), max U=16, T=500

        rpt #4
norm *
SACH GPR0

```



```
SPM 0

LACC HALF_PERIOD
    SUB GPRO
LDP #232
SACL CMPR3

RET    ; return
;End of routine inverter_I

;=====
; Routine Name: sinewave_gen.asm
; Originator   : Zhenyu Yu
;               Texas Instruments
;               DSP Digital Control Systems Applications
;-----
; Description:
;
; Generates reference sine wave for the output inverter stage
;
;-----
; Debug directives
;-----

        .def  RUN

        .bss one,1 ; +1
        .bss set_F,1; set F input,Q15, (0=0Hz,7FFFh-120Hz)
        .bss F_W,1 ; set F to angular speed ratio,Q6
        .bss S_W,1 ; set angular speed,Q5
        .bss min_W,1; lower limit on set W,Q5
        .bss Ta,1

        .bss S_U,1 ; set voltage,Q14
        .bss T_sample,1 ; sampling period,Q24

        .bss THETAH,1; THETA higher word,Q12
        .bss THETAL,1; THETA lower word
        .bss theta_r,1 ; rounded THETAH,Q12

        .bss theta_m,1 ; THETA mapped to 1st quadrant,Q12
        .bss theta_i,1 ; theta to index for sine table,Q8
        .bss SS,1 ; sin sign modification,Q0
        .bss SC,1 ; cos sign modification,Q0
        .bss index,1; index to sine table,Q0
        .bss SIN_1st,1 ; beginning of sin table
        .bss SIN_last,1 ; end of sin table
        .bss sin_theta,1 ; sin(THETA),Q14
        .bss cos_theta,1 ; cos(THETA),Q14
        .bss Ud,1 ; voltage Ud,Q12
        .bss Uq,1 ; voltage Uq,Q12

        .bss theta_60,1 ; 60,Q12
        .bss theta_90,1 ; 90,Q12
        .bss theta_120,1 ; 120,Q12
```





```

.bss theta_180,1 ; 180,Q12
.bss theta_240,1 ; 240,Q12
.bss theta_270,1 ; 270,Q12
.bss theta_300,1 ; 300,Q12
.bss theta_360,1 ; 360,Q12

.bss T1_periods,1 ; scaled Timer 1 period,Q5

        .bss  RUN,1

;-----
; Program parameters
;-----
; Low frequency sampling period Ts_l=100uS, Freq Fs_l = 10KHz
; In Q24 sampling period = Ts_l*2e+24 = 1678
; Scaled sampling period T_sample_ = 1678
;
T_sample_ .set 1678 ; Q24 (Use this when sinewave is generated in
; 10KHz sampling loop)

debug_data .set 7626h ; (65Hz,7FFFh)(60Hz,7626h)(45Hz,589ch)

; Set frequency to radian frequency conversion ratio
; 65*2*pi = 408.407045
; 7FFFh corresponds to 65Hz=408.407045 rad/sec
F_W_ .set 26138 ; Q6

; Minimum radian frequency
; min_F*2*pi=45*2*pi = 282.7433
; min_F=45Hz is the minimum frequency input
min_W_ .set 9048 ; Q5

;magnitude of ref voltage Uout
S_U_ .set 12000

; Conversion from theta to index for sine table:
; 360/(0.5pi)= ;229.1831181
theta_i_ .set 29335 ; Q7

.data
;-----
; Frequently used angles
;-----
*****
** The order between these angles must not be changed. **
*****
angles_ .WORD 010c1h ; pi/3: Q12
        .WORD 01922h ; pi/2: Q12
        .WORD 02183h ; 2*pi/3: Q12
        .WORD 03244h ; pi: Q12
        .WORD 04305h ; 4*pi/3: Q12
        .WORD 04b66h ; 3*pi/2: Q12
        .WORD 053c7h ; 5*pi/3: Q12
        .WORD 06488h ; 2*pi: Q12

```



sinwave\_gen:

```

    SETC SXM
;-----
; Obtain theta (phase of Uout) through 32 bit integration
;-----
    spm 0

        LDP    #6
    LTS_W    ; set W -> T: Q5
    MPY T_sample ; Q5*Q24=Q29
    PAC                      ; product -> ACC, ACCH in Q13
    SFR      ; Q12
    ADDH THETAH ; Q12
    ADDS THETAH ;
    SACH THETAH ; save
    SACL THETAH ;
    SUBH theta_360 ; compare with 2*pi
    BLEZ T_in_limit ; continue if within limit
    SACH THETAH ; mod(2*pi, THETA) if not
T_in_limit ZALH THETAH ; Zero ACCL and Load ACCH
    ADDS THETAH ; Add THETAH to ACC as a unsigned number
    ADD one,15 ;
    SACH theta_r ; round up to upper 16 bits

;-----
; Determine quadrant
;-----
    LACC one ; assume THETA (THETAH) is in quadrant 1
    SACL SS ; 1=>SS, sign of SIN(THETA)
    SACL SC ; 1=>SC, sign of COS(THETA)
    LACC theta_r ;
    SACL theta_m ; THETA=>theta_m
    SUB theta_90 ;
    BLEZ E_Q ; jump to end if 90>=THETA

        ; assume THETA (THETAH) is in quadrant 2
    splk #-1,SC ; -1=>SC
    LACC theta_180 ;
    SUB theta_r ; 180-THETA
    SACL theta_m ; =>theta_m
    BGEZ E_Q ; jump to end if 180>=THETA

        ; assume THETA (THETAH) is in quadrant 3
    splk #-1,SS ; -1=>SS
    LACC theta_r ;
    SUB theta_180 ; THETA-180
    SACL theta_m ; =>theta_m
    LACC theta_270 ;
    SUB theta_r ;
    BGEZ E_Q ; jump to end if 270>=THETA

        ; THETA (THETAH) is in quadrant 4

```



```
splk #1,SC ; 1=>SC
LACC theta_360 ;
SUB theta_r ;
SACL theta_m ; 360-THETAH=>theta_m
E_Q

;-----
; sin(theta), cos(theta)
;-----
lt theta_m ; Q12. Find index
mpy theta_i ; Q12*Q7=Q19
pac ; q19(32-bit)

sach index ; Q3. Make index an integer (Q0)
lacc index,13 ;
sach index ; right shift 3 bits => Q0

lac SIN_1st ; Look up sin
add index ;
tblr sin_theta ;
lac SIN_last ;
sub index ;
tblr cos_theta ;

LTSS ; Look up cos
MPY sin_theta ; modify sign: Q0*Q14=Q14
PAC ;
tag2: SACL sin_theta ; left shift 16 bits and save: Q14
LTSC ;
MPY cos_theta ; modify sin
PAC ;
SACL cos_theta ; left shift 16 bits and save:Q14

;-----
; Calcualte Ud & Uq
;-----

LTS_U ; set U -> T: Q14
MPY cos_theta ; set U*cos(THETA): Q14*Q14=Q28
PAC ; product -> ACC: Q28
SACH Ud,1 ; d component of ref Uout:Q13
MPY sin_theta ; set U*sin(THETA): Q14*Q14=Q28
PAC ; product -> ACC: Q28
SACH Uq,3 ; q component of ref Uout: Q15
RET ; return
;End of routine sinwave_gen
```



```
-----  
; Routine Name: init_batchcharge.asm  
; Originator   : Shamim Choudhury  
;       Texas Instruments  
;       DSP Digital Control Systems Applications  
-----  
; Description:  
;  
; Initializes battery charger stage variables and coefficients  
;  
-----  
init_batchcharge:  
  
    ldp #En0_abc ; set DP for ".bat_chg"  
        SPLK #04227H,KVB ;Q15,Scaling constant for VB  
        SPLK #0409aH,Vref_bc ;Q15,Ref charging volt  
  
        SPLK #0421dH,K_vbc ;Q25  
        SPLK #00520H,K0_vbc ;Q6  
        SPLK #0003fH,K1_vbc ;Q6  
        SPLK #004e0H,K2_vbc ;Q6  
        SPLK #038e2H,KU1_vbc ;Q14  
SPLK #01c72H,KU2_vbc ;Q16  
    SPLK #00000H,En0_vbc ;Q15  
    SPLK #00000H,En1_vbc ;Q15  
SPLK #00000H,En2_vbc ;Q15  
    SPLK #00000H,Unvbc_L_0 ;Q15  
    SPLK #00000H,Unvbc_H_0 ;Q15  
    SPLK #00000H,Un2vbc ;Q15  
  
SPLK #07fffH,K_abc ;Q14  
    SPLK #00134H,K0_abc ;Q14  
    SPLK #00123H,K1_abc ;Q14  
    SPLK #0ffefH,K2_abc ;Q14  
    SPLK #017f5H,KU1_abc ;Q14  
SPLK #05016H,KU2_abc ;Q15  
    SPLK #00000H,En0_abc ;Q15  
    SPLK #00000H,En1_abc ;Q15  
SPLK #00000H,En2_abc ;Q15  
    SPLK #00000H,Unabc_L_0 ;Q15  
    SPLK #00000H,Unabc_H_0 ;Q15  
    SPLK #00000H,Un2abc ;Q15  
  
Ret
```



```

;-----
; Routine Name: BatCharge_v.asm
; Originator   : Shamim Choudhury
;               Texas Instruments
;               DSP Digital Control Systems Applications
;-----
; Description:
;
; Implements battery charger voltage control algorithm
;
;-----
; Debug directives
  .def K0_vbc
  .def K1_vbc
      .def K2_vbc
      .def KU1_vbc
      .def KU2_vbc
  .def En0_vbc
  .def En1_vbc
      .def En2_vbc
  .def Unvbc_L_0
  .def Unvbc_H_0
      .def Un2vbc
      .def Vbatt
      .def Vref_bc

;-----
; Variables declaration
;-----
K_vbc   .usect ".bat_chg",1      ;compensator coefficient K
K0_vbc  .usect ".bat_chg",1;compensator coefficient K0
K1_vbc  .usect ".bat_chg",1;compensator coefficient K1
K2_vbc  .usect ".bat_chg",1      ;compensator coefficient K2
KU1_vbc .usect ".bat_chg",1;compensator coefficient Ku1
KU2_vbc .usect ".bat_chg",1;compensator coefficient Ku2
En0_vbc .usect ".bat_chg",1;Voltage error En0
En1_vbc .usect ".bat_chg",1;Voltage error En1
En2_vbc .usect ".bat_chg",1;Voltage error En2
Unvbc_L_0 .usect ".bat_chg",1;compensator output(lower 16-bit)
Unvbc_H_0 .usect ".bat_chg",1;compensator output(upper 16-bit)
Un2vbc  .usect ".bat_chg",1;compensator output Un2
Vbatt   .usect ".bat_chg",1      ;Battery terminal voltage

GPR0_vbc .usect ".bat_chg",1      ;General purpose register
GPR1_vbc .usect ".bat_chg",1      ;General purpose register
KVB      .usect ".bat_chg",1;Scaling constant for VB
          ;KVB=123/238=4227h(Q15)
          ;Used for converting VB from the range
          ;(-123V ~ +123V) to (-238V ~ +238V).
Vref_bc .usect ".bat_chg",1      ;Reference charging voltage
Vtestc  .usect ".bat_chg",1;Not used

```



```

;-----
BatCharge_v:

        SETC SXM

LDP #K0_vbc

        MAR *,AR0 ;Set AR0 as the auxiliary register

LAR AR0,#VB
LACC * ;Read ADC results for voltage VB
XOR #08000h
NEG          ;Invert polarity, since VB sense amplifier
            ;is in inverting configuration.
SUB #1
SACL GPR0_vbc ;Q15

;Change VB range from(-123V ~ +123V) to (-238V ~ +238V)
SPM 0
LT KVB      ;Q15
MPY GPR0_vbc ;Q15*Q15
PAC          ;ACC(Q30), 32-bit
NORM *      ;Q31
SACH GPR0_vbc ;Q15,

LAR AR0,#V_cap2
LACC *      ;Read V_cap2(0~7fffh <=> 0~238V)
ADD GPR0_vbc
SACL Vbatt ;Q15,(-238V ~ +238V)<=>(8000h ~ 7fffh)
            ;Vbatt = GPR0_vbc + V_cap2

;*****
;Subtract measured battery voltage from the reference voltage and
compute error.
;Apply compensation(G_vbc) on error to compute charging current command
;Unvbc_H_0.
;*****
        LACC Vref_bc ;q15
        SUB Vbatt ;Q15
            SACL En0_vbc ;Q15

;Calculate new current command Unvbc_H_0 based on error

SPM 0
LT      En1_vbc ;T<--En1_vbc(Q15),
MPY     K1_vbc ;P<--K1_vbc*En1_vbc(Q6*Q15=Q21),

            LTP En2_vbc ;T<--En2_vbc(Q15),P-->ACC(Q21)
DMOV    En1_vbc ;En1_vbc-->En2_vbc
MPY     K2_vbc ;P<--K2_vbc*En2_vbc(Q6*Q15=Q21),

LTS En0_vbc ;ACC(Q21)= K1_vbc*En1_vbc
            ;- K2_vbc*En2_vbc
            ;T<--En0_vbc(Q15)

```



```

DMOV     En0_vbc ;En0_vbc-->En1_vbc
MPY K0_vbc ;P<--K0_vbc*En0_vbc(Q6*Q15=Q21),

APAC          ;ACC(Q21)= K1_vbc*En1_vbc - K2_vbc*En2_vbc
              ;+ K0_vbc*En0_vbc

          NORM * ;ACC(Q22)
SACH GPR1_vbc ;GPR1_vbc(Q6) = K1_vbc*En1_vbc
              ;- K2_vbc*En2_vbc
              ;+ K0_vbc*En0_vbc

LT GPR1_vbc ;Q6
MPY K_vbc ;P<--K_vbc*GPR1_vbc(Q25*Q6=Q31),

LTP Un2vbc ;T<--Un2vbc(Q15),P-->ACC(Q31)
MPY KU2_vbc ;P<--KU2_vbc*Un2vbc(Q16*Q15=Q31),

LTD Unvbc_H_0 ;ACC(Q31)= K_vbc*(K1_vbc*En1_vbc
              ; - K2_vbc*En2_vbc
              ; + K0_vbc*En0_vbc)
; + KU2_vbc*Un2vbc
              ;T<--Unvbc_H_0(Q15),
;Unvbc_H_0-->Un2vbc(Q15)
SFR          ;ACC(Q30)
SFR          ;ACC(Q29)
MPY KU1_vbc ;P<--KU1_vbc*Unvbc_H_0(Q14*Q15=Q29)

APAC ;ACC(Q29)= K_vbc*(K1_vbc*En1_vbc - K2_vbc*En2_vbc
              ;+ K0_vbc*En0_vbc)
;+ KU2_vbc*Un2vbc + KU1_vbc*Unvbc_H_0
SACH GPR1_vbc ;GPR_vbc(Q13)= K_vbc*(K1_vbc*En1_vbc -
              ; - K2_vbc*En2_vbc
              ; + K0_vbc*En0_vbc)
;+ KU2_vbc*Un2vbc + KU1_vbc*Unvbc_H_0
SACL GPR0_vbc
LACC GPR1_vbc ;Q13
BCND Uvbc_lo_lmt,LT ;If current command is -ve,
;saturnate at -ve Umax

          LACC GPR1_vbc ;Q13
SUB #00925H
BCND Uvbc_hi_lmt,GEQ ;If current command is > 2A,
;saturnate max +ve current command to 2A
B done_lmt_Uvbc
Uvbc_lo_lmt
SPLK #0h,GPR1_vbc ;-ve Umax = 0
SPLK #0h,GPR0_vbc
B done_lmt_Uvbc

Uvbc_hi_lmt
SPLK #00925h,GPR1_vbc ;max +ve charging current = 2A
;=> +ve Umax = ((7fffh/Q15)/7A)*(2A)*(Q13)
;=00925h in Q13
SPLK #0000h,GPR0_vbc

```



done\_lmt\_Uvbc

```
ZALS GPR0_vbc
ADDH GPR1_vbc      ;ACC(Q29), 32-bit
SACH Unvbc_H_0,2  ;Q15, save new control output
SACL Unvbc_L_0
```

```
RET      ; return
```

;End of routine BatCharge\_v

```
=====
; Routine Name: BatCharge_i.asm
; Originator   : Shamim Choudhury
;               Texas Instruments
;               DSP Digital Control Systems Applications
;-----
; Description:
;
; Implements battery charger current controller and generates PWM3
; for Q3.
;-----
; Debug directives
.def K0_abc
.def K1_abc
        .def K2_abc
        .def KU1_abc
        .def KU2_abc
.def En0_abc
.def En1_abc
        .def En2_abc
.def Unabc_L_0
.def Unabc_H_0
        .def Un2abc
        .def Ibattc
        .def Iref_bc
.def GPR0_abc
.def GPR0_abc
;-----
; Variables declaration
;-----
K0_abc .usect ".bat_chg",1;Compensator coefficient K0
K1_abc .usect ".bat_chg",1;Compensator coefficient K1
K2_abc .usect ".bat_chg",1      ;Compensator coefficient K2
KU1_abc.usect ".bat_chg",1;Compensator coefficient Ku1
KU2_abc.usect ".bat_chg",1;Compensator coefficient Ku2
En0_abc.usect ".bat_chg",1;Current error En0
En1_abc.usect ".bat_chg",1;Current error En1
En2_abc.usect ".bat_chg",1;Current error En2
Unabc_L_0 .usect ".bat_chg",1;Compensator output(lower 16-bit)
Unabc_H_0 .usect ".bat_chg",1;Compensator output(upper 16-bit)
Un2abc .usect ".bat_chg",1;Compensator output Un2
Ibattc .usect ".bat_chg",1      ;Battery inductor current
K_abc .usect ".bat_chg",1 ;Not used
```





```

Iref_bc .usect ".bat_chg",1      ;Not used
GPR0_abc .usect ".bat_chg",1     ;General purpose register
GPR1_abc .usect ".bat_chg",1     ;General purpose register
Itestc .usect ".bat_chg",1;Not used
;-----
BatCharge_i:

        SETC SXM
        MAR *,AR2

        LDP #K0_abc
        LAR AR2,#Ib

                LACC      *      ;Read battery current sample from ADC
        XOR #08000h
        NEG      ;Invert polarity, since Ib sense amplifier
                ;is in inverting configuration.
        SUB #1
        SFL
        SACL Ibattc ;Q15

;-----
;Subtract charging current from reference current command and compute
;error.
;Apply compensation(G_abc) on error and then update compare value and
;ACTR to generate PWM3.
;-----

        LACC Unvbc_H_0 ;Q15
                sub      Ibattc      ;Q15
        SACL En0_abc ;Q15.

;Calculate new control output Unibc_H_0 based on error

        SPM 0
        LT Un2abc ;T <-- Un2abc(Q15)
        MPY   KU2_abc      ;P <-- KU2_abc*Un2abc(Q15*Q15=Q30)

        LTP Unibc_H_0 ;T <-- Unibc_H_0(Q15),P-->ACC(Q30)
        SFR      ;ACC(Q29)
        DMOV Unibc_H_0 ;Unibc_H_0 --> Un2abc(Q15)
        MPY KU1_abc      ;P <-- KU1_abc*Unibc_H_0(Q14*Q15=Q29)

        LTA   En2_abc ;ACC(Q29) <-- KU2_abc*Un2abc +
;+ KU1_abc*Unibc_H_0
                ;T <-- En2_abc,
        MPY   K2_abc      ;P <-- K2_abc*En2_abc(Q14*Q15=Q29)

                LTD En1_abc      ;ACC(Q29) <-- KU2_abc*Un2abc
                ;+ KU1_abc*Unibc_H_0
                ;+ K2_abc*En2_abc
                ;T <-- En1_abc, En1_abc --> En2_abc
        MPY K1_abc      ;P <-- K1_abc*En1_abc(Q14*Q15=Q29)

```



```

LTD En0_ibc      ;ACC(Q29) <-- KU2_ibc*Un2ibc
                  ;+ KU1_ibc*Unibc_H_0
                  ;+ K2_ibc*En2_ibc + K1_ibc*En1_ibc
                  ;T <-- En0_ibc, En0_ibc --> En1_ibc
MPY K0_ibc      ;P <-- K0_ibc*En0_ibc(Q14*Q15=Q29)
APAC           ;ACC(Q29) <-- KU2_ibc*Un2ibc
                  ;+ KU1_ibc*Unibc_H_0
                  ;+ K2_ibc*En2_ibc + K1_ibc*En1_ibc
                  ;+ K0_ibc*En0_ibc

SACH GPR1_ibc ;Q13
SACL GPR0_ibc

LACC GPR1_ibc ;Q13
    BCND Uibc_lo_lmt,LT ;If -ve, saturate at -ve Umax

    LACC GPR1_ibc ;Q13
SUB #01f00H
    BCND Uibc_hi_lmt,GEQ ;If maxed out, saturate at +ve Umax
    B done_lmt_Uibc
Uibc_lo_lmt
    SPLK #0h,GPR1_ibc ;-ve Umax = 0
    SPLK #0h,GPR0_ibc
    B done_lmt_Uibc

Uibc_hi_lmt
    SPLK #01f00h,GPR1_ibc ;+ve Umax = 0.96875
    SPLK #0000h,GPR0_ibc

done_lmt_Uibc
    ZALS GPR0_ibc
    ADDH GPR1_ibc ;ACC(Q29), 32-bit
    SACH Unibc_H_0,2 ;Q15, save new control output
    SACL Unibc_L_0

;Convert Q15 value to an absolute Q0 for use in Compare reg.
    LAR AR2,#T1PER
    LT Unibc_H_0 ; Q15
    MPY * ; P = Unibc_H_0 * T1PER
    PAC ; ACC in Q15, 32-bit
    SACH GPR0_ibc,1 ; Q0, GPR0_ibc = Unibc_H_0* T1PER

    LACC *
    SUB GPR0_ibc ;ACC = T1PER - T1PER*Unibc_H_0
    SACL GPR0_ibc ;Save

```



```
;Update as follows, if polarity of PWM4 is AH. If PWM4 is FL then  
;branch  
;to no_update_bc.  
;During battery boost operation PWM4 is AH and PWM3 is FL(Q3 off).  
;During battery charge operation PWM3 is AH and PWM4 is FL(Q4 off).
```

```
LDP    #232  
        BIT ACTR,BIT7  
BCND no_update_bc,NTC ;  
  
LAR AR2,#ACTR  
LACC *  
AND #0F0Fh  
SACL *           ;Configure PWM3,4=FL  
  
SPLK #00001011110000111b,COMCON  
;           ||||||||||||||||  
;           FEDCBA9876543210  
SPLK #10001011110000111b,COMCON  
  
RPT #14  
NOP  
  
SPLK #00000011110000111b,COMCON  
;           ||||||||||||||||  
;           FEDCBA9876543210  
SPLK #10000011110000111b,COMCON  
LACC *  
OR #0000000000100000b  
SACL *           ;Configure PWM3=AH and PWM4=FL  
LAR AR2,#GPR0_ibc  
LACC *           ;Load CMPR2 value  
  
no_update_bc:  
    SACL CMPR2    ;CMPR2 = T1PER - T1PER*Unibc_H_0  
    RET          ; Return  
  
;End of routine BatCharge_i
```



```
=====
; Routine Name: Z_xing_pll.asm
; Originator   : Zhenyu Yu
;               Texas Instruments
;               DSP Digital Control Systems Applications
;-----
; Description
;
; This code implements zero crossing detection and phase calculation
; algorithm for the UPS input voltage.
;
;-----
; Global variables
;-----

    .def supply_cycle ; Supply cycle flag: 1 - positive;
                                ; 0 - negative
    .def supply_phase ; supply phase

;-----
; Variable definitions (in B1 page 0)
;-----
Vs_cur .usect ".xingpll",1; D0, current sample, +-1.0 - +-2.5
Vs_old .usect ".xingpll",1; D0, last sample
sign_cur .usect ".xingpll",1; D15, sign of current sample
sign_old .usect ".xingpll",1; D15, sign of old sample
supply_cycle .usect ".xingpll",1; D15, supply cycle
cycle_time .usect ".xingpll",1; D-4, time in a cycle
T_cycle .usect ".xingpll",1; D-4, time for half cycle
supply_phase .usect ".xingpll",1; D3, supply phase: 0-2*pi
supply_freq .usect ".xingpll",1; D7, supply freq: 0-120Hz
supply_omega .usect ".xingpll",1; D?, supply angular freq: 0-
; 2*pi*120 rad/sec
K_omega .usect ".xingpll",1;
K_f .usect ".xingpll",1;
sampling_pr .usect ".xingpll",1; D-14, sampling period

sampling_pr_ .set 21475 ; D-14, 0.000040*2**29
x_thrshld_ .set 0400h ; 0-xing threshold

temp0 .usect ".xingpll",1; temporary storage
temp1 .usect ".xingpll",1; temporary storage

;-----
Z_xing_pll:
    mar *,AR2 ; set ARP
    ldp #Vs_cur; set DP

    lacc Vs_cur ; save old sample
    sacl Vs_old ;

    lacc sign_cur ; save old sign
    sacl sign_old ;
```



```
SETC SXM
lar ar2,#Vs; set AR
lacc * ;
xor #08000h; Convert back to bipolar and D0 format
NEG ; Correct the polarity

SUB #1
sac1 Vs_cur ; Save current supply voltage sample
BCND non_z_samp,NEQ ; check to see if sample is zero
RET ;

non_z_samp BIT supply_cycle,BIT0 ; test supply cycle flag
BCND add_thrsh,TC ; add threshold if pos cycle
sub #x_thrshld_ ; sub threshold if neg cycle
Bcontinue_pll ;
add_thrsh add #x_thrshld_ ;
continue_pll
sac1 GPR0 ;

bcnd cur_zero,EQ ; to cur_zero if current sample is zero

lacc #0 ; extract sign of current sample
BIT GPR0,BIT15
bcnd cur_pos,NTC ; current sample is positive

cur_neglacc #1 ; current sample is negative
cur_possac1 sign_cur ;

sub sign_old ; check to see if there is zero crossing
bcnd n_xing,EQ ; to n_xing (no zero crossing) if equal

lt Vs_old ;det zero xing instant T0=Yn*Ts/(Yn-Yn+1)
;D0
mpy sampling_pr ; D0*D-14=D-13, Yn*Ts
pac ;
abs ;
sach temp0,1 ; D-14
lacc Vs_old ; Yn-Yn+1
sub GPR0 ;
abs ;
sac1 temp1 ;
lacl temp0 ;
rpt #15 ; D-14/D0=D15-14-0=D1, T0=Yn*Ts/(Yn-Yn+1)
subc temp1 ;
sac1 temp0 ;

lacc cycle_time ; update length of time for half cycle
add temp0 ;
sac1 T_cycle ;

lacc cycle_time ; update cycle time
add sampling_pr ;
sac1 cycle_time ;
```



```
    lacc sign_old ; update cycle flag
cycle_flg: sacl supply_cycle ;

    bcnd n_start,EQ ; check to see if start of new period
    lacc sampling_pr ; reset cycle time based on
                                ; t=Ts-T0 if neg
    sub temp0 ;
    sacl cycle_time ;
n_start

    lacl K_f ; calculate f=0.5/T_cycle
    rpt #15
    subc T_cycle ;
    sacl supply_freq ;
    lt supply_freq ; 2*pi*f
    mpy K_omega ;
    pac
    sach supply_omega ;

    lt supply_omega ; update phase
    mpy cycle_time ;
    pac ;
    sach supply_phase ;
    ret

; Run simplified algorithm when sample is zero
cur_zero lacc sign_old ; set old sign to avoid reporting extra
; zero crossing
    xor #1 ;
    sacl sign_cur ;

    lacc cycle_time ;
    add sampling_pr ;
    sacl T_cycle ; update length of time for half cycle
    sacl cycle_time ; update cycle time

    lacc sign_old ; update cycle flag
    sacl supply_cycle ;

    bcnd n_start2,EQ ; check to see if start of new period
    splk #0,cycle_time ; reset cycle_time if yes
n_start2

    lacl K_f ; calculate f=0.5/T_cycle
    rpt #15
    subc T_cycle ;
    sacl supply_freq ;
    lt supply_freq ; calculate w=2*pi*f
    mpy K_omega ;
    pac
    sach supply_omega ;
    splk #0,supply_phase ; reset phase
    ret
```



```

; Run simplified algorithm when there is not zero crossing
n_xing lacc cycle_time ; update cycle time
    add sampling_pr ;
    sac1 cycle_time ;

    lt cycle_time ; update phase
    mpy supply_omega ;
    pac ;
    sach supply_phase ;
    ret ; return

; -----
; Initialization of routine
; -----
K_f_ .set 1000 ;
K_omega_ .set 1000 ;

init_xingpll:
    ldp #sampling_pr ; set DP
    splk #sampling_pr_,sampling_pr
    splk #K_f_,K_f
    splk #K_omega_,K_omega
    splk #0,sign_cur;
    splk #0,Vs_cur;
    ret
;=====
; Routine Name: init_pfc.asm
; Originator : Shamim Choudhury
; Texas Instruments
; DSP Digital Control Systems Applications
;-----
; Description:
;
; Initializes PFC stage variables and coefficients
;
;-----
init_pfc:
    ldp #En0_ipfc ; set DP for pfc stage

;current loop compensator coefficients and variables

    SPLK #00000H,En0_ipfc ;Q15
        SPLK #00000H,En1_ipfc ;Q15
        SPLK #00000H,En2_ipfc ;Q15
        SPLK #00000H,Unipfc_L_0 ;Q15
        SPLK #00000H,Unipfc_H_0 ;Q15
        SPLK #00000H,Un2_ipfc ;Q15

        SPLK #04e9cH,K_ipfc ;Q14
    SPLK #0d06cH,Ku2_ipfc ;Q16(-0.18585)
    SPLK #0a9faH,K2_ipfc ;Q9(-43.011)
    SPLK #00304H,K1_ipfc ;Q6
    SPLK #0058dH,K0_ipfc ;Q5

```



;voltage loop PI coefficients and variables

```
SPLK #07fffH,K0_v1pfc ;Q15
      SPLK #08100H,K1_v1pfc ;Q15

      SPLK #00000H,En1_v1pfc ;Q15
      SPLK #00000H,Unv1pfc_L_0;Q15
      SPLK #00000H,Unv1pfc_H_0;Q15

SPLK #07fffH,K0_v2pfc ;Q15
      SPLK #08100H,K1_v2pfc ;Q15

      SPLK #00000H,En1_v2pfc ;Q15
      SPLK #00000H,Unv2pfc_L_0;Q15
      SPLK #00000H,Unv2pfc_H_0;Q15
```

RET

```
=====
; Routine Name: PFCBoost_i.asm
; Originator : Shamim Choudhury
; Texas Instruments
; DSP Digital Control Systems Applications
;
; Description:
;
; This code implements the PFC stage current control algorithm and
; generates PWM1 and PWM2 signals for the switches Q1 and Q2
; respectively.
;
;
; Debug directives
```

```
.def Ku2_ipfc,1
      .def K0_ipfc,1
.def K1_ipfc,1
.def K2_ipfc,1
.def K_ipfc,1

.def En0_ipfc,1
.def En1_ipfc,1
      .def En2_ipfc,1
.def Unipfc_L_0,1
.def Unipfc_H_0,1
      .def Un2_ipfc,1
```

```
=====
; Variables in B1 page 0
;=====
```

```
.bss Ku2_ipfc,1
      .bss K0_ipfc,1
.bss K1_ipfc,1
.bss K2_ipfc,1
.bss K_ipfc,1

.bss En0_ipfc,1
```





```

.bss En1_ipfc,1
        .bss En2_ipfc,1
.bss Unipfc_L_0,1
.bss Unipfc_H_0,1
        .bss  Un2_ipfc,1

.bss Iin,1

;-----
PFCBoost_i:

    MAR *,AR2
    LDP #En0_ipfc

    LACC Is
    XOR #8000h
    NEG
    SUB #1
        SACL Iin ;Q15

;-----
;Subtract input current from reference current command and compute
;error.
;Apply compensation Gc(s) on error and then update compare value and
;ACTR to
;generate PWM1 and PWM2.
;-----

    SETC SXM

    LACC Iref_pfc ;q15
        sub  Iin      ;Q15
    SACL En0_ipfc ;Store error(Q15).

;Calculate new control output Un based on error

    SPM 0

    ZALS Unipfc_L_0 ;ACC = Un-1(Q25,32bit)
    ADDH Unipfc_H_0

    LT Un2_ipfc ;TREG <-- Un-2(Q9)
    DMOV Unipfc_H_0 ;Un-1 --> Un-2 (Q9)
        MPY  Ku2_ipfc ;PREG <-- KU2*Un-2(Q16*Q9=Q25)
        ;Ku2_ipfc in q16

        LTA  En2_ipfc ;Q15,TREG <-- En-2,
        ;ACC <-- Un-1 + KU2*Un-2 (Q25)
    DMOV En1_ipfc ;Q15,En-1 --> En-2
    sfr ;acc(q24)

    MPY K2_ipfc ;q15*q9, PREG <-- K2*En-2
        ;K2_ipfc in q9
    apac ;Q24

```



```

SACH GPR1    ;Q8
SACL GPR0

spm 1

        LT    GPR1    ;Q8
        MPY   K_ipfc  ;Q8*Q14,
;PREG <-- K(Un-1 + KU2*Un-2 + K2*En-2)
        PAC   ;Q23, ACC <-- K(Un-1 + KU2*Un-2 - K2*En-2
        ; + k1*En-1 + K0*En-0)

LT En1_ipfc  ;Q15
DMOV En0_ipfc ;Q15, En0 --> En-1

sfr      ;acc(Q22)

MPY K1_ipfc;Q15*Q6, PREG <-- K1*En-1
apac    ;ACC(Q22)

LT En0_ipfc ;Q15, TREG <-- En-0, En-0 --> En-1,
        ;ACC <-- Un-1 + KU2*Un-2 - K2*En-2 + k1*En-1

sfr      ;acc(q21)

MPY K0_ipfc;Q15*Q5, PREG <-- K0*En-0
APAC    ;ACC <-- Un-1 + KU2*Un-2 - K2*En-2
        ; + k1*En-1 + K0*En-0
        ;acc(q21)

        spm 0

        SACH GPR1    ;Q5
SACL GPR0

LACC GPR1
ADD #07ffH
        BCND  U_lo_lmt,LEQ      ;If maxed out, saturate at max -ve U

        LACC  GPR1              ;else keep current value of U
SUB   #07ffH
        BCND  U_hi_lmt,GEQ      ;If maxed out, saturate at max +ve U
        B     done_lmt_U

U_lo_lmt
SPLK #0f801h,GPR1 ;max -ve U =-64(Q5)
SPLK #00000h,GPR0
        B     done_lmt_U

U_hi_lmt
SPLK #07ffh,GPR1 ;max +ve U =64(Q5)
SPLK #0000h,GPR0

```



done\_lmt\_U

```
ZALS GPR0 ;ACC in q21, 32 bit
ADDH GPR1
```

```
rpt #3
SFL ;ACC(Q25)
```

```
SACH Unipfc_H_0 ;Q9
SACL Unipfc_L_0
```

;Convert Q9 value to an absolute Q0 for use in Compare reg.

SPM 3

```
LTUnipfc_H_0 ; (Q9)
MPY HALF_PERIOD ; P = Unipfc_H_0 * T1PER/2
PAC ; ACC = (Unipfc_H_0/64) * T1PER/2
; ACCH in Q-7, ACCL in Q9.
SACH GPR2,7
; So GPR2 is in Q0.
; GPR2 = U* T1PER/2
spm 0
```

end\_pfc\_i:

```
ldp #supply_cycle; set DP for supply cycle flag
LACC supply_cycle
BCND pos_supply, NEQ
```

neg\_supply

ld\_cmpr

```
LAR AR2, #ACTR
LDP #232
BIT ACTR, BIT3
BCND no_update1, NTC ;
```

;update with the following if polarity of PWM2 is AH. If PWM2 is FL  
;then branch to no\_update1

```
LACC * ;Read ACTR
AND #0ff0h ;Configure PWM1 and PWM2 as FL
SACL * ;Update ACTR
```

```
SPLK #0000101110000111b, COMCON
; | | | | | | | | | | | | | | | | | |
; | | | | | | | | | | | | | | | | | |
SPLK #1000101110000111b, COMCON
```

RPT #14



```

NOP

SPLK #0000001110000111b,COMCON
;          ||||||||||||||||
;          FEDCBA9876543210
SPLK #1000001110000111b,COMCON
no_update1

LACC * ;Read ACTR
OR #0002h ;Configure PWM1 as AH and PWM2 as FL
SACL * ;Update ACTR

LAR AR2,#GPR2
LACC * ; ACC= GPR2 = U* T1PER/2
LAR AR2,#HALF_PERIOD
ADD * ; ACC = T1PER/2 + U* T1PER/2

SACL CMPR1 ; CMPR1 = T1PER/2 + U* T1PER/2
B DONE_ICTRL
;=====
pos_supply

LAR AR2,#ACTR
LDP #232
BIT ACTR,BIT1
BCND no_update2,NTC ;
;update with the following if polarity of PWM1 is AH. Otherwise branch
;to no_update2

LACC * ;Read ACTR
AND #0ff0h ;Configure PWM1 and PWM2 as FL
SACL * ;Update ACTR

SPLK #0000101110000111b,COMCON
;          ||||||||||||||||
;          FEDCBA9876543210
SPLK #1000101110000111b,COMCON

RPT #14
NOP

SPLK #0000001110000111b,COMCON
;          ||||||||||||||||
;          FEDCBA9876543210
SPLK #1000001110000111b,COMCON

no_update2
LACC * ;Read ACTR
OR #0008h ;Configure PWM1 as FL and PWM2 as AH
SACL * ;Update ACTR

LAR AR2,#HALF_PERIOD
LACC * ;ACC = T1PER/2
LAR AR2,#GPR2
SUB * ; ACC = T1PER/2 - U* T1PER/2
```



```
SACL CMPR1 ;CMPR1 = T1PER/2 - U* T1PER/2
DONE_ICTRL:
    RET ; return

;End of routine PFCBoost_i
;=====
; Routine Name: PFCBoost_v.asm
; Originator : Shamim Choudhury
; Texas Instruments
; DSP Digital Control Systems Applications
;-----
; Description:
;
; This code implements the PFC stage dc bus capacitors(C1 and C2)
; voltage control algorithms and finally generates the current command
; required for the current control loop.
;
;-----
; Debug directives
.def K0_v1pfc
.def K1_v1pfc
.def En0_v1pfc
.def En1_v1pfc
.def Unv1pfc_H_0
.def Unv1pfc_L_0
.def K0_v2pfc
.def K1_v2pfc
.def En0_v2pfc
.def En1_v2pfc
.def Unv2pfc_H_0
.def Unv2pfc_L_0

;-----
; Variables in B1 page 0
;-----
.bss K0_v1pfc,1
.bss K1_v1pfc,1
.bss En0_v1pfc,1
.bss En1_v1pfc,1
.bss Unv1pfc_H_0,1
.bss Unv1pfc_L_0,1

.bss K0_v2pfc,1
.bss K1_v2pfc,1
.bss En0_v2pfc,1
.bss En1_v2pfc,1
.bss Unv2pfc_H_0,1
.bss Unv2pfc_L_0,1

.bss Vpos,1
.bss Vneg,1
.bss max_Vref,1
.bss V_cap1,1
.bss V_cap2,1
```



```
.bss V_ref,1
.bss Iref_pfc,1

rmp_dly_max .set 25
;-----
PFCBoost_v:

    SETC SXM

    ldp #supply_cycle

    LACC Vneg
    XOR #8000h
    SACL V_cap2

    LACC Vpos
        XOR #8000h
    NEG
    SUB #1
    SACL V_cap1

        LACC supply_cycle ;1 => +ve half cycle
    ;0 => -ve half cycle
        BCND POS_CNTRL, NEQ ;Check for +ve and -ve half cycle
                                ;of the input supply voltage

;Based on the error voltage calculate the control output Univ2 for the
;negative half cycle of the input supply voltage

NEG_CNTRL:

        LACC V_ref
    sub V_cap2
    SACL En0_v2pfc ;Q15

    SPM 1

    ZALS Unv2pfc_L_0 ;q31, ACC = Un-1
    ADDH Unv2pfc_H_0

    LT En1_v2pfc ;TREG <-- En1 (Q15)
    MPY K1_v2pfc ;q15*q15,PREG <-- K1*En1

    LTD En0_v2pfc ;q31,ACC <-- Un-1 + K1*En-1,
        ;TREG<--En0, En0-->En1
    MPY K0_v2pfc ;q15*q15,PREG <-- K0*En0,
    APAC ;q31,ACC <-- Un-1 + K1*En-1 + K0*En-0

    SACH Unv2pfc_H_0 ;q15,ACC --> Un-0
    SACL Unv2pfc_L_0

    SPM 0
    LACC Unv2pfc_H_0
```



```

        BCND  SAT_M_IV2,LT      ;If maxed out, saturate at max -ve U
        LACC  Unv2pfc_H_0      ;else keep current value of U
        SUB   #7000h
        BCND  SAT_P_IV2, GEQ    ;If maxed out, saturate at max +ve U
        LACC  Unv2pfc_H_0      ;else keep current value of U
        B     FWD_IV2

SAT_M_IV2
        SPLK  #0,Unv2pfc_H_0
        SPLK  #0, Unv2pfc_L_0
        B     FWD_IV2

SAT_P_IV2
        SPLK  #7000h,Unv2pfc_H_0
        SPLK  #0, Unv2pfc_L_0

FWD_IV2:

        spm 1
        lt Vs_cur   ; Q15
        mpy Unv2pfc_H_0 ;Q30 (Q15*Q15)
        pac        ;q31
        sach Iref_pfc ;Q15
        spm 0
        BDONE_N_CNTRL

;Calculate new control output Univl based on error

POS_CNTRL:

        LACC  V_ref      ;
        sub V_cap1      ;
        SACL  En0_vlpfc ;Store error(Q15).

        SPM 1

        ZALS  Unvlpfc_L_0 ;q31,ACC = Un-1
        ADDH  Unvlpfc_H_0
        LT  En1_vlpfc ;TREG <-- En1 (Q15)
        MPY  K1_vlpfc ;q15*q15,PREG <-- K1*En1,
        LTD  En0_vlpfc ;q31,ACC <-- Un-1 + K1*En-1
        MPY  K0_vlpfc ;q15*q15,PREG <-- K0*En0,
        APAC ;q31,ACC <-- Un-1 + K1*En-1 + K0*En-0

        SACH  Unvlpfc_H_0 ;q15,ACC --> Un-0
        SACL  Unvlpfc_L_0

        SPM 0

        LACC  Unvlpfc_H_0
        BCND  SAT_M_IV1,LT      ;If maxed out, saturate at max -ve U

```



```

        LACC  Unvlpfc_H_0      ;else keep current value of U
        SUB  #7000h
        BCND  SAT_P_IV1, GEQ   ;If maxed out, saturate at max +ve U

        LACC  Unvlpfc_H_0      ;else keep current value of U
        B     FWD_IV1

SAT_M_IV1
        SPLK  #0,Unvlpfc_H_0
        SPLK  #0, Unvlpfc_L_0
        B     FWD_IV1

SAT_P_IV1
        SPLK  #7000h,Unvlpfc_H_0
        SPLK  #0, Unvlpfc_L_0

FWD_IV1:

    spm 1
    lt Vs_cur ; Q15
    mpy Unvlpfc_H_0 ; Q30 (Q15*Q15)
    pac ; q31
    sach Iref_pfc ; Q15
    spm 0

DONE_N_CNTRL:

;Slowly increase the capacitor reference voltage 'V_ref' to the
;specified maximum reference voltage 'max_Vref_trgt'.

        lacc max_Vref
        sacl V_ref

        lacc max_Vref_trgt
        sub max_Vref
        bcnd fd_end, EQ

        lacc rmp_dly_cnt
        add #1
        sacl rmp_dly_cnt
        sub #rmp_dly_max
        bcnd fd_end2, LT

CHNG_VREF:
        lacc max_Vref_trgt
        sub max_Vref
        bcnd inc_Vref, GT

        Bfd_end

inc_Vref lacc max_Vref
        add #1
        sacl max_Vref
        sub #6000h
```





```
        bcnd fd_end, LEQ
            splk #6000h, max_Vref

fd_end:splk #0, rmp_dly_cnt

fd_end2

        RET
;End of routine PFCBoost_v

;=====
; Routine Name:  init_batboost.asm
; Originator   :  Shamim Choudhury
;               :  Texas Instruments
;               :  DSP Digital Control Systems Applications
;-----
; Description:
;
; Initializes battery boost stage variables and coefficients
;
;-----
init_batboost:
    ldp #En0_ibb; set DP for ".boost"

            SPLK #02000H,K_ibb;battery current gain K(Q15)

;Battery Voltage Boost stage current loop compensator coefficients and
;variables

    SPLK #01c1aH,K0_ibb ;Q15
            SPLK #0e560H,K1_ibb ;Q15
            SPLK #070f7H,Ku1_ibb ;Q15

            SPLK #00000H,En0_ibb ;Q15
            SPLK #00000H,En1_ibb ;Q15
            SPLK #00000H,Unibb_L_0 ;Q15
            SPLK #00000H,Unibb_H_0 ;Q15

;Battery Voltage Boost stage voltage loop PI coefficients and variables

    SPLK #07fffH,K0_vbb ;Q15 format (k0=1)
            SPLK #08042H,K1_vbb ;Q15 format (K1=-0.998)

            SPLK #00000H,En1_vbb ;Q15
            SPLK #00000H,Unvbb_L_0 ;Q15
            SPLK #00000H,Unvbb_H_0 ;Q15

    SPLK #0,dly_cnt_vbb
    splk #07000h,Unvbb_trgt
    splk #0000h,max_unvbbp
            splk #0000h,max_unvbbm

    ret
```



```
=====
; Routine Name: BatBoost_i.asm
; Originator   : Shamim Choudhury
;               Texas Instruments
;               DSP Digital Control Systems Applications
;-----
; Description:
;
; Implements boost stage current control algorithm and generates PWM4
; for the switch Q4.
;-----
; Debug directives
  .def K0_abb
  .def K1_abb
      .def Kul_abb
  .def En0_abb
  .def En1_abb
  .def Unibb_H_0
  .def Unibb_L_0
  .def GPR0_abb
;-----
; Variable definitions (in B1P1)
;-----
K_abb .usect ".boost",1 ;battery current gain K(=51fh in Q15)
K0_abb .usect ".boost",1 ;PI coefficient K0 for boost
K1_abb .usect ".boost",1 ;PI coefficient K1 for batt boost
Kul_abb .usect ".boost",1 ;PI coefficient Kul for batt boost
En0_abb .usect ".boost",1 ;Error En0 for batt boost
En1_abb .usect ".boost",1 ;Error En1 for batt boost
Unibb_L_0 .usect ".boost",1 ;Control output Unibb_L_0 for batt boost
Unibb_H_0 .usect ".boost",1 ;Control output Unibb_H_0 for batt boost

Ibatt .usect ".boost",1
GPR0_abb .usect ".boost",1

MAX_POS_Unibb .set 01000h ;Q12(Max Pos Uni=1)
MAX_NEG_Unibb .set 0000h ;Q12(Max Neg Uni=0)
;-----

BatBoost_i:
  MAR *,AR2
  LDP #Unibb_H_0 ;set DP for ".boost" variables

  SETC SXM

  LAR AR2,#Ib
  LACC *
  XOR #08000h
  NEG
  SACL Ibatt ;Q12
```



```

SPM 1
    lt Ibatt    ;Q12
    mpy K_ibb   ;Q15
    PAC        ;Q28(32 bit)
            SACH GPR0_ibb ;Q12

    lacc Unvbb_H_0 ;Q12
            SUB GPR0_ibb ;Q12
    SACL En0_ibb ;Store error(Q12).

;Calculate new control output Unibb based on error

    LT Unibb_H_0 ;T = Un-1(Q12)
    MPY Ku1_ibb  ;Q15*Q12
    PAC          ;ACC=Ku1*Un1,Q28

    LT En1_ibb ;TREG <-- En1 (Q12)
    MPY K1_ibb ;P<- K1*En1, Q15*Q12

    LTD En0_ibb ;ACC<- Ku1*Un1 + K1*En-1, Q28
    MPY K0_ibb  ;P<- K0*En0,Q15*Q12
            ;K0 in Q15

    APAC        ;ACC <-- Ku1*Un1 + K1*En-1 + K0*En-0, Q28

    SACH Unibb_H_0 ;ACC --> Un-0(Q12)
    SACL Unibb_L_0

SPM 0

LACC Unibb_H_0
    BCND SAT_MINUS_ibb,LT ;If maxed out, saturate at max -ve U
    LACC Unibb_H_0        ;else keep current value of U
    SUB    #01000H
    BCND SAT_PLUS_ibb,GEQ ;If maxed out, saturate at max +ve U
    B      FWD_ibb

SAT_MINUS_ibb
    SPLK    #MAX_NEG_Unibb,Unibb_H_0
    SPLK    #0, Unibb_L_0
    B      FWD_ibb

SAT_PLUS_ibb
    SPLK    #MAX_POS_Unibb,Unibb_H_0
    SPLK    #0, Unibb_L_0

FWD_ibb:

;Convert Q12 value to an absolute Q0 for use in Compare reg.
    LAR AR2,#T1PER
    LT Unibb_H_0 ; Q12
    MPY * ; PREG = Unibb_H_0 * T1PER,
            ; PREGH in Q-4 and PREGL in Q12

```



```
PAC    ; PREG -> ACC(32bit)
      ; ACC = Unibb_H_0 * T1PER
SACH GPR0_ibb,4 ; GPR0_ibb = Unibb_H_0* T1PER
LACC *
SUB GPR0_ibb ; ACC = T1PER - T1PER*Unibb_H_0
SACL GPR0_ibb ; save CMPR2 value

LDP    #232
      BIT ACTR,BIT5
BCND no_update3,NTC ;

;update with the following if polarity of PWM3 is AH. If PWM3 is FL
;then branch to no_update3

LAR AR2,#ACTR
LACC *
AND #0F0Fh
SACL *           ;configure PWM3,4=FL

SPLK #0000101110000111b,COMCON
;           ||||||||||||||||
;           FEDCBA9876543210
SPLK #1000101110000111b,COMCON

RPT #14
NOP

SPLK #0000001110000111b,COMCON
;           ||||||||||||||||
;           FEDCBA9876543210
SPLK #1000001110000111b,COMCON
LACC *
OR #0000000010000000b
SACL *           ;Configure PWM3=FL and PWM4=AH
LAR AR2,#GPR0_ibb
LACC *           ;Load CMPR2 value

no_update3:
SACL CMPR2 ;CMPR2 = T1PER - T1PER * Unibb_H_0

RET    ; return

;End of routine BatBoost_i
```



```
=====
; Routine Name: BatBoost_v.asm
; Originator   : Shamim Choudhury
;               Texas Instruments
;               DSP Digital Control Systems Applications
;-----
; Description:
;
; Implements boost stage voltage control algorithm
;-----
; Debug directives
; .def K0_vbb
; .def K1_vbb
; .def En0_vbb
; .def En1_vbb
; .def Unvbb_H_0
; .def Unvbb_L_0
; .def GPR0_vbb
;-----
; Variable definitions
;-----
K0_vbb .usect ".boost",1 ;PI coefficient K0 for batt boost
K1_vbb .usect ".boost",1 ;PI coefficient K1 for batt boost
En0_vbb .usect ".boost",1 ;Error En0 for batt boost
En1_vbb .usect ".boost",1 ;Error En1 for batt boost
Unvbb_L_0 .usect ".boost",1 ;Control output Unvbb_L_0 for batt boost
Unvbb_H_0 .usect ".boost",1 ;Control output Unvbb_H_0 for batt boost

GPR0_vbb .usect ".boost",1

dly_cnt_vbb .usect ".boost",1
Unvbb_trgt .usect ".boost",1
max_unvbbp .usect ".boost",1
max_unvbbm .usect ".boost",1

dly_max_vbb .set 10

MAX_POS_Unvbb .set 07000h ;Q15(Max Pos Unv=0.625)
MAX_NEG_Unvbb .set 09000h ;Q15(Max Neg Unv=-0.625)
;-----
BatBoost_v:
    SETC SXM
    MAR *,AR0 ;Set AR0 as the auxiliary register

    LDP #6
    LACC Vneg ;
    XOR #8000h
    SACL V_cap2

    LACC Vpos ;
    XOR #8000h

    NEG
    SUB #1
    SACL V_cap1
```



```

LACC V_cap1
SFR
LAR AR0, #GPR0_vbb
SACL *
LACC V_cap2
SFR
ADD *
SACL V_bus

SPM 1

lacc V_ref
sub V_bus

LDP #En0_vbb ;set DP for ".boost"

SACL En0_vbb ;Store error(Q15).

;Calculate new control output Unv based on error

ZALS Unvbb_L_0 ;ACC = Un-1(Q31)
ADDH Unvbb_H_0

LT En1_vbb ;TREG <-- En1 (Q15)
MPY K1_vbb ;P<- K1*En1, Q15*Q15

LTD En0_vbb ;ACC<- Un-1 + K1*En-1, Q31,
MPY K0_vbb ;P<- K0*En0, Q15*Q15
;K0 in Q15

APAC ;ACC <-- Un-1 + K1*En-1 + K0*En-0, Q31

SACH Unvbb_H_0 ;ACC --> Un-0(Q15)
SACL Unvbb_L_0

SPM 0

LACC Unvbb_H_0

ADD max_unvbbp
BCND SAT_MINUS_vbb,LT ;If maxed out, saturate at max -ve U
LACC Unvbb_H_0 ;else keep current value of U
sub max_unvbbp

BCND SAT_PLUS_vbb,GEQ ;If maxed out, saturate at max +ve U
B FWD_vbb

SAT_MINUS_vbb

lacc max_unvbbm
sac1 Unvbb_H_0
SPLK #0, Unvbb_L_0
B FWD_vbb

```



```
SAT_PLUS_vbb
    lacc max_unvbbp
    sac1 Unvbb_H_0
        SPLK #0, Unvbb_L_0
FWD_vbb:

        lacc Unvbb_trgt
        sub max_unvbbp
    bcnd fd_end_vbb, EQ

        lacc dly_cnt_vbb
    add #1
    sac1 dly_cnt_vbb
        sub #dly_max_vbb

    bcnd fd_end_vbb2, LT

CHNG_Unvbb:
    lacc Unvbb_trgt
        sub max_unvbbp
    bcnd inc_unvbb, GT

    Bfd_end_vbb

inc_unvbb lacc max_unvbbp
    add #1
        sac1 max_unvbbp

    neg
    sac1 max_unvbbm
    neg
    sub #7000h
    bcnd fd_end_vbb, LEQ
        splk #7000h, max_unvbbp
        splk #9000h, max_unvbbm

fd_end_vbb: splk #0, dly_cnt_vbb

fd_end_vbb2

    RET
```

;End of routine BatBoost\_v

```
;-----
; sine table for theta from 0 to 90 per every 0.25 degrees
;-----
```

```
SIN_TABLE_      ; sin table
.WORD 0      ; D1
.WORD 71
.WORD 143
.WORD 214
.WORD 286
.WORD 357
```



.WORD 429  
.WORD 500  
.WORD 572  
.WORD 643  
.WORD 715  
.WORD 786  
.WORD 857  
.WORD 929  
.WORD 1000  
.WORD 1072  
.WORD 1143  
.WORD 1214  
.WORD 1285  
.WORD 1357  
.WORD 1428  
.WORD 1499  
.WORD 1570  
.WORD 1641  
.WORD 1713  
.WORD 1784  
.WORD 1855  
.WORD 1926  
.WORD 1997  
.WORD 2068  
.WORD 2139  
.WORD 2209  
.WORD 2280  
.WORD 2351  
.WORD 2422  
.WORD 2492  
.WORD 2563  
.WORD 2634  
.WORD 2704  
.WORD 2775  
.WORD 2845  
.WORD 2915  
.WORD 2986  
.WORD 3056  
.WORD 3126  
.WORD 3196  
.WORD 3266  
.WORD 3336  
.WORD 3406  
.WORD 3476  
.WORD 3546  
.WORD 3616  
.WORD 3686  
.WORD 3755  
.WORD 3825  
.WORD 3894  
.WORD 3964  
.WORD 4033  
.WORD 4102  
.WORD 4171  
.WORD 4240





.WORD 4310  
.WORD 4378  
.WORD 4447  
.WORD 4516  
.WORD 4585  
.WORD 4653  
.WORD 4722  
.WORD 4790  
.WORD 4859  
.WORD 4927  
.WORD 4995  
.WORD 5063  
.WORD 5131  
.WORD 5199  
.WORD 5266  
.WORD 5334  
.WORD 5402  
.WORD 5469  
.WORD 5536  
.WORD 5604  
.WORD 5671  
.WORD 5738  
.WORD 5805  
.WORD 5872  
.WORD 5938  
.WORD 6005  
.WORD 6071  
.WORD 6138  
.WORD 6204  
.WORD 6270  
.WORD 6336  
.WORD 6402  
.WORD 6467  
.WORD 6533  
.WORD 6599  
.WORD 6664  
.WORD 6729  
.WORD 6794  
.WORD 6859  
.WORD 6924  
.WORD 6989  
.WORD 7053  
.WORD 7118  
.WORD 7182  
.WORD 7246  
.WORD 7311  
.WORD 7374  
.WORD 7438  
.WORD 7502  
.WORD 7565  
.WORD 7629  
.WORD 7692  
.WORD 7755  
.WORD 7818  
.WORD 7881



.WORD 7943  
.WORD 8006  
.WORD 8068  
.WORD 8130  
.WORD 8192  
.WORD 8254  
.WORD 8316  
.WORD 8377  
.WORD 8438  
.WORD 8500  
.WORD 8561  
.WORD 8621  
.WORD 8682  
.WORD 8743  
.WORD 8803  
.WORD 8863  
.WORD 8923  
.WORD 8983  
.WORD 9043  
.WORD 9102  
.WORD 9162  
.WORD 9221  
.WORD 9280  
.WORD 9339  
.WORD 9397  
.WORD 9456  
.WORD 9514  
.WORD 9572  
.WORD 9630  
.WORD 9688  
.WORD 9746  
.WORD 9803  
.WORD 9860  
.WORD 9917  
.WORD 9974  
.WORD 10031  
.WORD 10087  
.WORD 10143  
.WORD 10199  
.WORD 10255  
.WORD 10311  
.WORD 10366  
.WORD 10422  
.WORD 10477  
.WORD 10531  
.WORD 10586  
.WORD 10641  
.WORD 10695  
.WORD 10749  
.WORD 10803  
.WORD 10856  
.WORD 10910  
.WORD 10963  
.WORD 11016  
.WORD 11069



.WORD 11121  
.WORD 11174  
.WORD 11226  
.WORD 11278  
.WORD 11330  
.WORD 11381  
.WORD 11433  
.WORD 11484  
.WORD 11535  
.WORD 11585  
.WORD 11636  
.WORD 11686  
.WORD 11736  
.WORD 11786  
.WORD 11835  
.WORD 11885  
.WORD 11934  
.WORD 11982  
.WORD 12031  
.WORD 12080  
.WORD 12128  
.WORD 12176  
.WORD 12223  
.WORD 12271  
.WORD 12318  
.WORD 12365  
.WORD 12412  
.WORD 12458  
.WORD 12505  
.WORD 12551  
.WORD 12597  
.WORD 12642  
.WORD 12688  
.WORD 12733  
.WORD 12778  
.WORD 12822  
.WORD 12867  
.WORD 12911  
.WORD 12955  
.WORD 12998  
.WORD 13042  
.WORD 13085  
.WORD 13128  
.WORD 13170  
.WORD 13213  
.WORD 13255  
.WORD 13297  
.WORD 13338  
.WORD 13380  
.WORD 13421  
.WORD 13462  
.WORD 13502  
.WORD 13543  
.WORD 13583  
.WORD 13623



.WORD 13662  
.WORD 13702  
.WORD 13741  
.WORD 13780  
.WORD 13818  
.WORD 13856  
.WORD 13894  
.WORD 13932  
.WORD 13970  
.WORD 14007  
.WORD 14044  
.WORD 14081  
.WORD 14117  
.WORD 14153  
.WORD 14189  
.WORD 14225  
.WORD 14260  
.WORD 14295  
.WORD 14330  
.WORD 14364  
.WORD 14399  
.WORD 14433  
.WORD 14466  
.WORD 14500  
.WORD 14533  
.WORD 14566  
.WORD 14598  
.WORD 14631  
.WORD 14663  
.WORD 14694  
.WORD 14726  
.WORD 14757  
.WORD 14788  
.WORD 14819  
.WORD 14849  
.WORD 14879  
.WORD 14909  
.WORD 14938  
.WORD 14968  
.WORD 14996  
.WORD 15025  
.WORD 15053  
.WORD 15082  
.WORD 15109  
.WORD 15137  
.WORD 15164  
.WORD 15191  
.WORD 15218  
.WORD 15244  
.WORD 15270  
.WORD 15296  
.WORD 15321  
.WORD 15346  
.WORD 15371  
.WORD 15396



.WORD 15420  
.WORD 15444  
.WORD 15468  
.WORD 15491  
.WORD 15515  
.WORD 15537  
.WORD 15560  
.WORD 15582  
.WORD 15604  
.WORD 15626  
.WORD 15647  
.WORD 15668  
.WORD 15689  
.WORD 15709  
.WORD 15729  
.WORD 15749  
.WORD 15769  
.WORD 15788  
.WORD 15807  
.WORD 15826  
.WORD 15844  
.WORD 15862  
.WORD 15880  
.WORD 15897  
.WORD 15914  
.WORD 15931  
.WORD 15948  
.WORD 15964  
.WORD 15980  
.WORD 15996  
.WORD 16011  
.WORD 16026  
.WORD 16041  
.WORD 16055  
.WORD 16069  
.WORD 16083  
.WORD 16096  
.WORD 16110  
.WORD 16123  
.WORD 16135  
.WORD 16147  
.WORD 16159  
.WORD 16171  
.WORD 16182  
.WORD 16193  
.WORD 16204  
.WORD 16214  
.WORD 16225  
.WORD 16234  
.WORD 16244  
.WORD 16253  
.WORD 16262  
.WORD 16270  
.WORD 16279  
.WORD 16287



.WORD 16294  
.WORD 16302  
.WORD 16309  
.WORD 16315  
.WORD 16322  
.WORD 16328  
.WORD 16333  
.WORD 16339  
.WORD 16344  
.WORD 16349  
.WORD 16353  
.WORD 16358  
.WORD 16362  
.WORD 16365  
.WORD 16368  
.WORD 16371  
.WORD 16374  
.WORD 16376  
.WORD 16378  
.WORD 16380  
.WORD 16382  
.WORD 16383  
.WORD 16383  
.WORD 16383

```
=====
; PHANTOM interrupts
; Description: Phantom int services.
=====
PHANTOMA EINT      ; Got a EV Group A phantom interrupt
  RET              ;
PHANTOMB EINT      ; Got a EV Group A phantom interrupt
  RET              ;
PHANTOM5 EINT      ; Got a Level 5 phantom interrupt
  RET              ;
PHANTOMEINT        ; Got other phantom interrupt
  RET              ;
```



## TI Contact Numbers

### INTERNET

*TI Semiconductor Home Page*

[www.ti.com/sc](http://www.ti.com/sc)

*TI Distributors*

[www.ti.com/sc/docs/distmenu.htm](http://www.ti.com/sc/docs/distmenu.htm)

### PRODUCT INFORMATION CENTERS

#### *Americas*

Phone +1(972) 644-5580

Fax +1(972) 480-7800

Email [sc-infomaster@ti.com](mailto:sc-infomaster@ti.com)

#### *Europe, Middle East, and Africa*

Phone

Deutsch +49-(0) 8161 80 3311

English +44-(0) 1604 66 3399

Español +34-(0) 90 23 54 0 28

Français +33-(0) 1-30 70 11 64

Italiano +33-(0) 1-30 70 11 67

Fax +44-(0) 1604 66 33 34

Email [epic@ti.com](mailto:epic@ti.com)

#### *Japan*

Phone

International +81-3-3344-5311

Domestic 0120-81-0026

Fax

International +81-3-3344-5317

Domestic 0120-81-0036

Email [pic-japan@ti.com](mailto:pic-japan@ti.com)

#### *Asia*

Phone

International +886-2-23786800

Domestic

Australia 1-800-881-011

TI Number -800-800-1450

China 10810

TI Number -800-800-1450

Hong Kong 800-96-1111

TI Number -800-800-1450

India 000-117

TI Number -800-800-1450

Indonesia 001-801-10

TI Number -800-800-1450

Korea 080-551-2804

Malaysia 1-800-800-011

TI Number -800-800-1450

New Zealand 000-911

TI Number -800-800-1450

Philippines 105-11

TI Number -800-800-1450

Singapore 800-0111-111

TI Number -800-800-1450

Taiwan 080-006800

Thailand 0019-991-1111

TI Number -800-800-1450

Fax 886-2-2378-6808

Email [tiasia@ti.com](mailto:tiasia@ti.com)

TI is a trademark of Texas Instruments Incorporated.

## **IMPORTANT NOTICE**

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

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

**CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.**

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty, or endorsement thereof.

Copyright © 1999 Texas Instruments Incorporated