Chapter 7

DSP-BASED IMPLEMENTATION OF DC-DC BUCK-BOOST CONVERTERS

7.1 Introduction

In a large number of industrial applications, it is required to convert a dc voltage to a different dc voltage level, often with a regulated output. To perform this task, a dc-dc converter is needed. A dc-dc converter directly converts a dc voltage of one level to another. It can be used to step-down (buck), or step-up (boost) a dc voltage source. In this chapter, the DSP-based control of a buck-boost, a specific type of dc-dc converter, is explained.

7.2 Converter Structure

The buck-boost converter has the structure shown in Fig. 7.1. The principle of operation is that when the transistor T is turned on, the input voltage V_{in} is applied across the inductor L and the current i_L in the inductor rises. Then the transistor is turned off. The current in the inductor must continue to flow somehow, and consequently finds its path through the load resistor R, and back to the inductor through the diode D. This discharges the inductor, and the current through it decreases. The capacitor C filters the output voltage ripple. The description given in the above is with the continuous conduction mode, meaning the inductor current never goes discontinuous. The continuous mode will be discussed further in the next section.

This converter has two dominant characteristics: the output voltage is always negative with respect to the input voltage and the output voltage may be higher or lower than the input voltage. This is why this converter may also be referred to as a step-up/step-down converter.

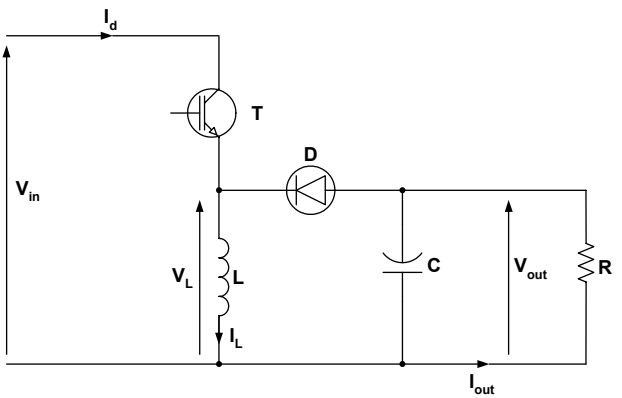

Figure 7.1 Buck-boost converter structure.

7.3 Continuous Conduction Mode

The input and output voltages are related by the following equation:

$$V_{out} = -\frac{d}{1-d} \times V_{in} \tag{7.1}$$

In this equation, d is the transistor or switch duty cycle. Figure 7.2 shows the switching pattern command to turn on or off, which must be fed to the transistor for proper operation of the buck-boost converter.

Figure 7.2 Transistor switching pattern.

Obviously, the duty cycle may vary only from 0 to 1. The resulting values for the converter voltage gain are:

$$d = 0 \Rightarrow G = \frac{V_{out}}{V_{in}} = 0 \tag{7.2}$$

$$d = 1 \Rightarrow G = \frac{V_{out}}{V_{in}} = -\infty \tag{7.3}$$

The theoretical gain range achievable is potentially very large. Practically, it is limited by the parasitic characteristics of the converter. In addition, it is often desirable to keep the duty cycle between 0.1 (10%) and 0.9 (90%) for practical engineering considerations. The relationship between the converter duty cycle and its gain, shown in Fig. 7.3, is non-linear.

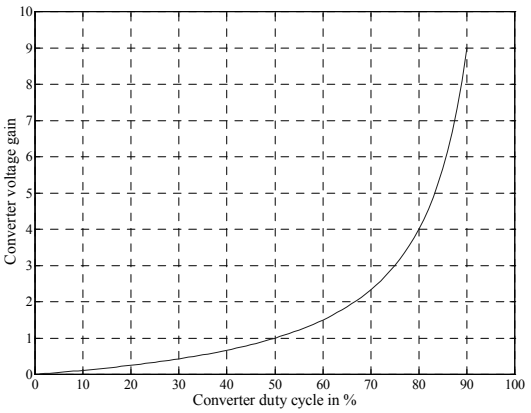

Figure 7.3 Converter voltage gain versus converter duty cycle.

7.4 Discontinuous Conduction Mode

The switching results in a cyclic current increase and decrease in the inductor. This current ripple has a non-negligible influence on the operation of the converter. If during the switching period T_S (shown in Fig. 7.2) the current never goes to zero, then the converter is said to operate in continuous conduction mode.

However, if the current does go to zero at any time, then the conduction is said to be discontinuous. In discontinuous conduction mode, the voltage gain of the converter is not solely a function of the duty cycle, but also of the output current. An example of a discontinuous conduction current waveform is shown in Fig. 7.4.

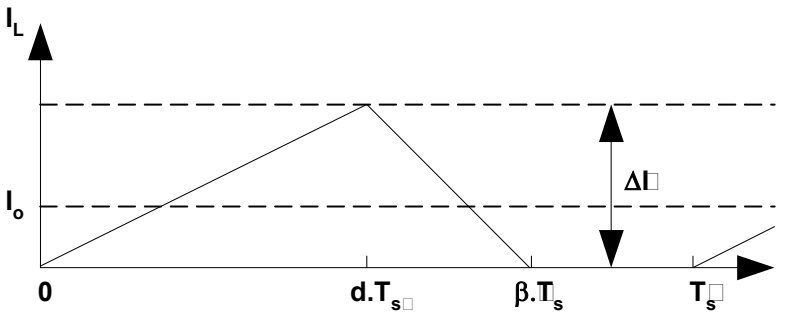

Figure 7.4 Discontinuous conduction mode current waveform.

7.5 Connecting the DSP to the Buck-Boost Converter

To fully control the buck-boost converter voltage and current with a DSP, one digital output and two analog inputs are required from the DSP. Figure 7.5 shows a conceptual connection diagram between the DSP and converter. The DSP will output a PWM switching waveform to the converter. The DSP will also receive information of the instantaneous current and voltage from the load via the analog to digital converter inputs. The following subsections describe the circuits necessary for interfacing the DSP to the dc-dc converter.

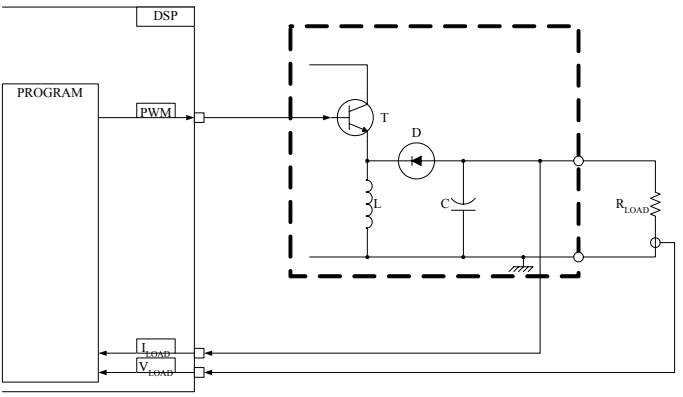

Figure 7.5 Physical implementation.

7.5.1 *Gate Driver*

The gate driver for this example is shown in Fig. 7.6 and is an integrated driver; it includes an opto-isolator, NPN transistor, PNP transistor, and the necessary logic to control them, all within an integrated package. Only the addition of two resistors is required to complete the gate driver circuit. The transistor that is being driven here is a MOSFET.

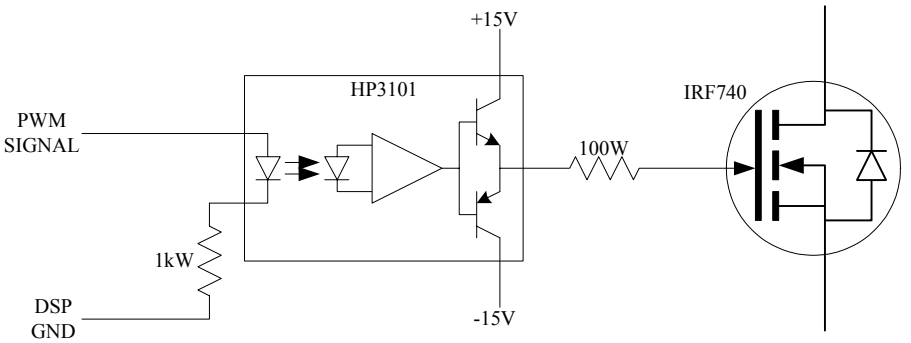

Figure 7.6 Gate driver circuit schematic.

7.5.2 *Current Sensor*

Current measurement can be performed using a shunt resistor in series with the output. This solution is more adapted to sensing small currents than a Hall-effect sensor and is also less expensive. The voltage across the 1Ω shunt resistor shown in Fig. 7.7 is buffered by a non-inverting amplifier, which provides infinite input impedance for the ADC input of the DSP. Due to the topology of the dc-dc converter, the output voltage is negative and must therefore be inverted. Because the shunt resistor is of a low value, the voltage across it will be small and must be amplified. A variable gain inverting amplifier provides for both these needs. The variable gain of the amplifier is used to adjust the gain of the sensed shunt resistor

voltage signal. The output of the amplifier is connected to an opto-isolator, which changes the signal path from electrical to optical, then back to an electrical signal. The optical transmission provides the necessary galvanic insulation between the power side of the converter and the DSP. This isolation is necessary because the DSP is a very sensitive device, while the converter is a major source of voltage surges and interferences. The operational amplifier feeds the input, or luminescent diode, of the opto-isolator. The output of the opto-isolator feeds into the DSP analog input. A variable collector resistor is used to set the offset voltage of the ADC input.

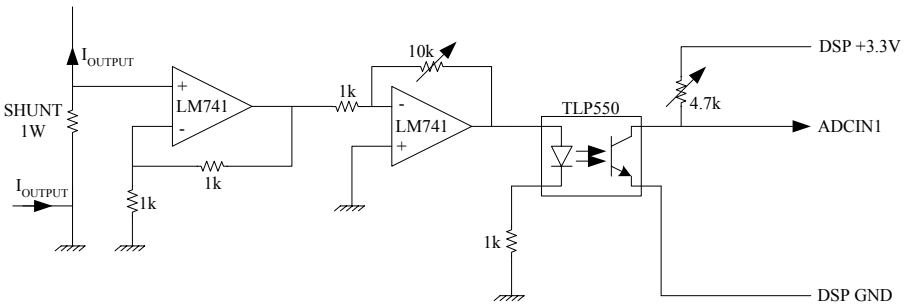

Figure 7.7 Current sensor circuit schematic.

7.5.3 Voltage Sensor

The voltage sensor uses the same circuit as the current sensor, but with a few differences. The output voltage of the converter is directly measurable and is directly fed into the infinite impedance buffer. The inverting amplifier is also slightly different in that it uses a 1kΩ resistor instead of a 10kΩ resistor. The difference is that while the current signal has to be amplified, the voltage level of the converter output must be attenuated to match the acceptable voltage range of the ADC input.

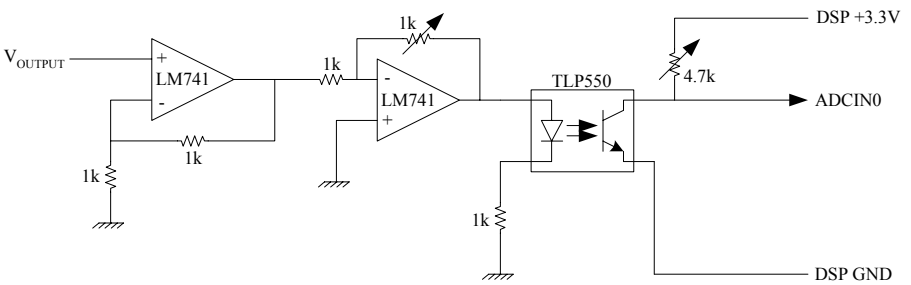

Figure 7.8 Voltage sensor circuit schematic.

7.6 Controlling the Buck-Boost Converter

The controller of a buck-boost converter usually has two objectives:

- Controlling the output voltage to a predetermined value
- Protecting the converter by limiting the output current to a predetermined value

Obviously, simply regulating the output voltage is the normal mode for the controller. If the load is such that the converter output voltage causes the current to go beyond the limit, then the controller must also control the converter to prevent the current from exceeding the maximum limit. Maintaining the current below the maximum is necessary in order to keep the converter and load from overheating. Because current regulation is necessary for safety, it must have the highest priority over all the other tasks in the control system, including the voltage regulation. This means that under any variation of the load, the current will be kept below its maximum.

The proposed control scheme has the following properties:

- The voltage will be regulated using a closed-loop PI regulator.

- The current will be checked every switching cycle, and if its value is above the limit, then the voltage regulation will be suspended. This will be achieved by setting the error signal to zero, which will disable the proportional action and disable the integration of the voltage error. The integrator is decremented by 1 every cycle in order to smoothly bring the voltage to a value, which will keep the current at its limit.

- The current regulation is effective if the current is only slightly above the limit. It is not effective against sudden surges such as a short circuit. If the current is above two times the maximum, then the controller will reset the PWM generation (thus shutting down the transfer of power from the source to the load) and reset the integrator. This will cause the converter restart from zero voltage in voltage regulation mode. Since the voltage is very low, the current will be slightly beyond the current limit, thus causing the controller to enter the current regulation mode. This is effective because output filter capacitor will be quickly discharged by a short circuit, and thus the output voltage will be zero. The voltage necessary for keeping the current below the limit in a short circuit condition is very low (in the order of a few mV), so the recovery from a short circuit should happen quickly.

Regulating the output voltage can be achieved easily by means of an integral regulator in a negative feedback loop. The integral regulator outputs a command of gain for the converter. For optimum performance, this command should be linearized; that is, converted into a useful duty cycle using the following equation:

$$d = \frac{Gain}{Gain + 1} \tag{7.4}$$

However, this is not mandatory because the negative feedback and the integral regulator ultimately ensure the convergence of the output voltage toward the reference. In addition, the DSP is very fast in reacting to variations in the system. Equation (7.4) is theoretical and does not take into account the parasitic elements of the system, which make the voltage gain a different function of the duty cycle. The theoretical voltage gain as a function of the duty cycle and the actual gain are plotted in Fig. 9. One sees that the two curves differ slightly. A look-up table would be the most accurate and simplest way to linearize the function. The voltage regulation is described using the block diagram shown in Fig. 7.10.

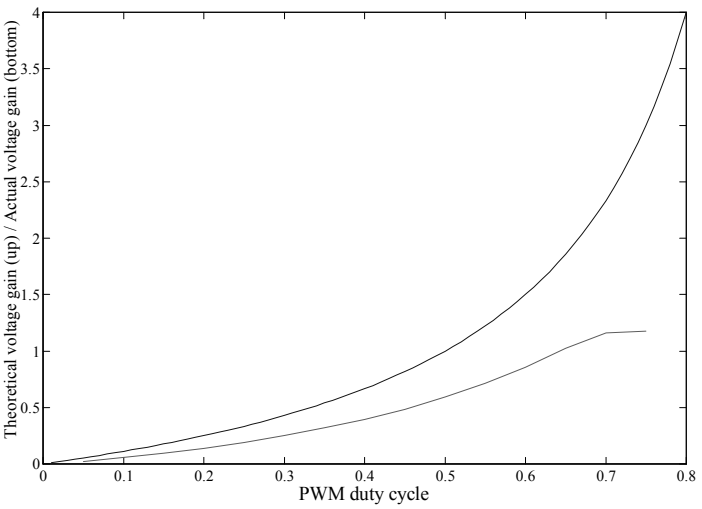

Figure 7.9 Divergence between theoretical and actual voltage gains.

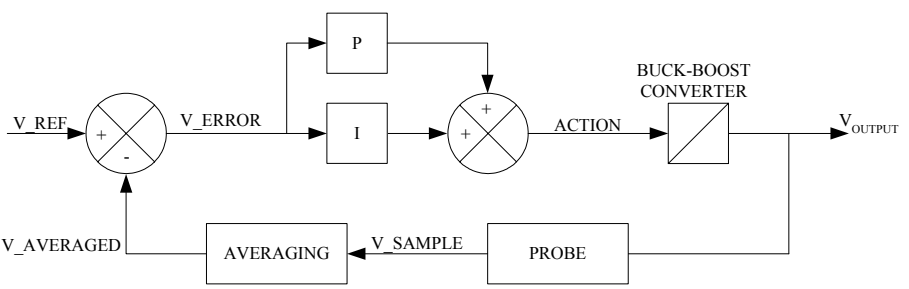

Figure 7.10 Voltage regulation block diagram.

This block diagram will be calculated once every switching cycle, which is the maximum speed at which parameters may be updated. Calculating the regulation more often would be useless because the actuator, the PWM generator, would not react until the next cycle.

The block diagram shown in Fig. 7.11 is implemented within the voltage regulation code in two sequences: current regulation and regulation reset.

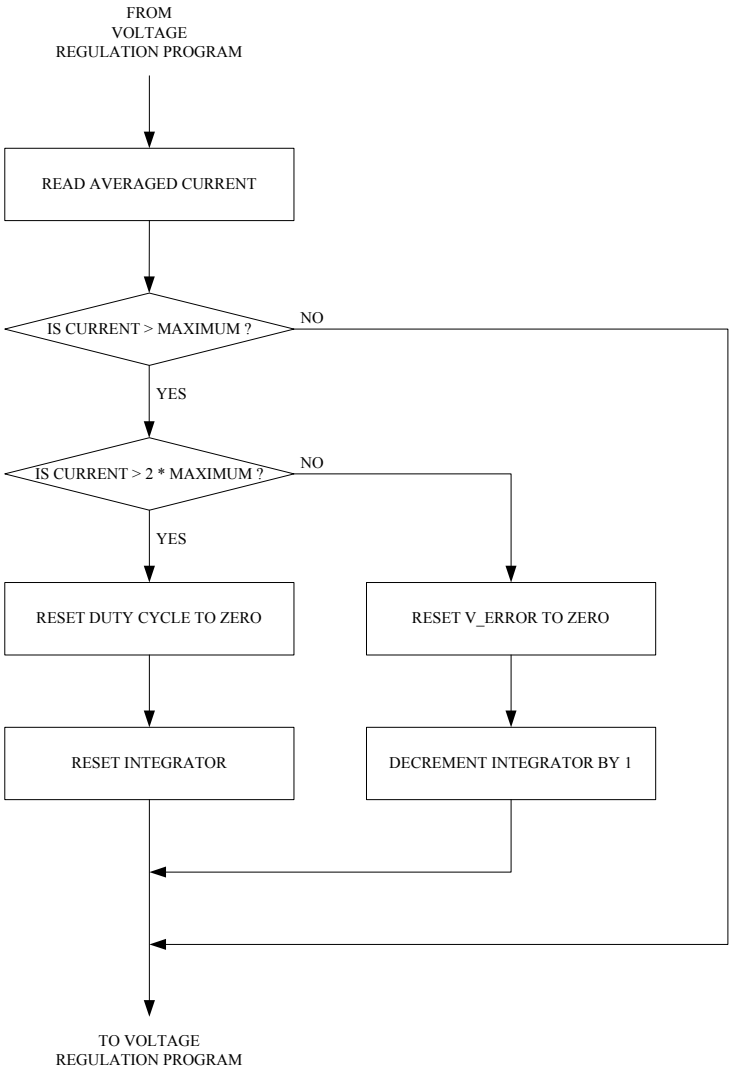

Figure 7.11 Current regulation algorithm.

Figure 7.12 illustrates the flow chart of the program developed in this chapter. Notice that several loops are used in the program.

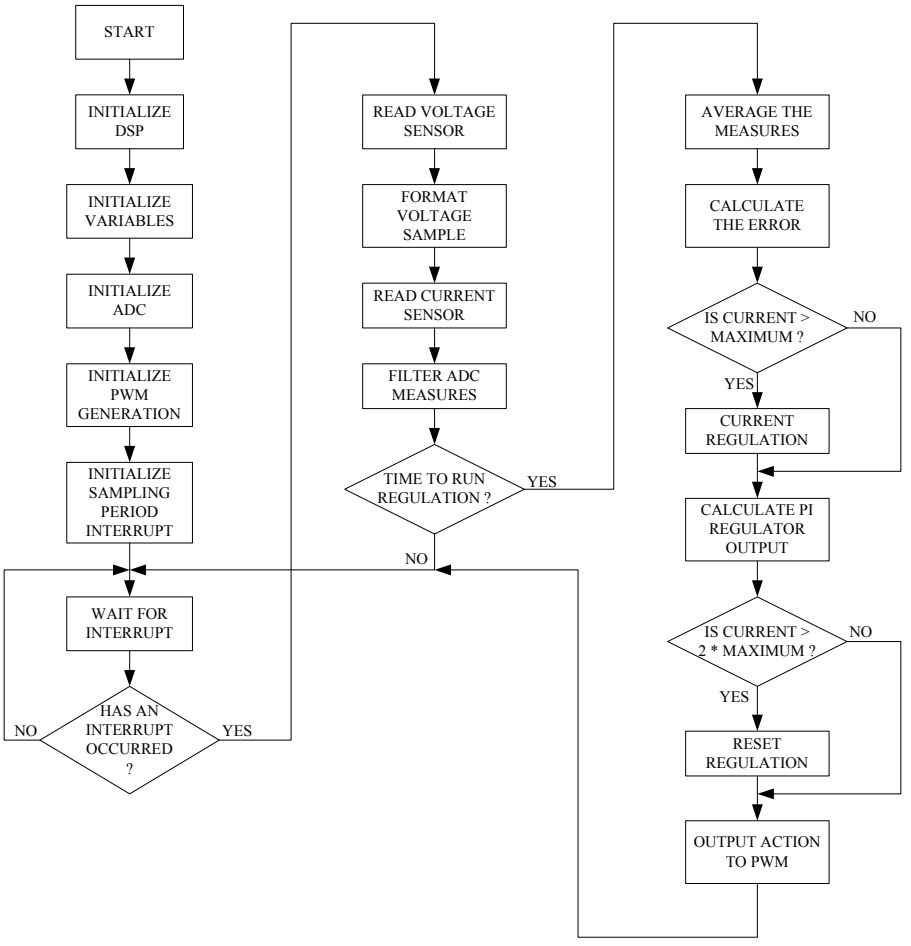

Figure 7.12 General program flow-chart.

7.7 Main Assembly Section Code Description

7.7.1 Variables Initialization

The block of code below initializes the defined variables with constants.

```
LDP     #06h
SPLK    #0900h,   V_OFFSET      ;Voltage probe offset * 2^6
SPLK    #0880h,   I_OFFSET      ;Current probe offset * 2^6
SPLK    #0000h,   V_SAMPLE      ;Voltage sample
SPLK    #0000h,   I_SAMPLE      ;Current sample
SPLK    #0000h,   V_SUM         ;Voltage sum
SPLK    #0000h,   I_SUM         ;Current sum
SPLK    #0000h,   V_AVERAGED    ;Voltage averaged
```

```
SPLK    #0000h,   I_AVERAGED    ;Current averaged
SPLK    #0138h,   V_REF         ;Voltage reference=5V=5000d/16d
SPLK    #0000h,   ACTION        ;Action output by the regulator
SPLK    #00FAh,   I_LIMIT       ;Current limit FAh = 250d= 25mA
SPLK    #0000h,   V_ERROR       ;Error of voltage regulation
SPLK    #0000h,   INTEGRAL      ;Regulation integrator
SPLK    #0000h,   INT_CNT       ;Interrupt counter
```

7.7.2 Initialization of the ADC

This code initializes the Analog-to-Digital Converter hardware of the LF2407 that performs the analog current and voltage sensing.

```
LDP     #ADCTRL1>>7h            ;Set data page corresponding to
                                ;ADCcontrol registers
SPLK    #0100000000000000b, ADCTRL1
                                ;Reset ADC
SPLK    #0011000000000000b, ADCTRL1
                                ;Set ADC for Bit 6=0 start-stop
                                ;mode
SPLK    #0001h,   MAXCONV       ;Set for 2 conversions (2
                                ;channels)
SPLK    #0010h,   CHSELSEQ1     ;Set for conversion on channel 1
                                ;and 0
SPLK    #0100000000000000b, ADCTRL2
                                ;Reset sequencer
SPLK    #0000000100000000b, ADCTRL2
                                ;Enables ADC to be started by an
                                ;event(timer 2 period here)
```

7.7.3 Initialization of GP Timer 1 for PWM Generation

This code sets the parameters of the PWM waveform generation. The period is set for a 1kHz carrier frequency. The duty cycle is set to a near zero value.

```
LDP     #0E1h                   ;Set data page corresponding to
                                ;GPIO pin registers
SPLK    #0FFFFh,  MCRA          ;Set GPIO pins for primary
                                ;function (IO)
LDP     #GPTCONA>>7h            ;Set data page corresponding to
                                ;general purpose timer control
                                ;register
SPLK    #00000h,  T1CNT         ;Reset timer 1 counter
SPLK    #3FFFh,   T1PR          ;Set timer 1 period to ~= 1ms
SPLK    #3FF0h,   T1CMPR        ;Set duty cycle
```

7.7.4 Sampling Period Interrupt Initialization

This code initializes GP Timer 2 and the interrupt operation for the Timer 2 period interrupt.

```
        SPLK    #00000h,  T2CNT         ;Reset timer 2 counter
        LACL    T1PR                    ;Load timer 1 (PWM) period
                                        ;register
        SACL    T2PR,     4             ;Divide by 16 (right shift by 4
                                        ;bits) and store in timer 2
                                        ;period register
        SPLK    #0000010001000010b, GPTCONA
                                        ;Set general purpose timer
                                        ;control register for: Bit 10,9
                                        ;= 10 start ADC upon timer 2
                                        ;period occurrence, Bit 6=1
                                        ;enables  timer  1  compare  output
                                        ;for PWM generation, Bit 2,1=10
                                        ;sets output pin polarity high
        SPLK    #1000100001000010b, T1CON
                                        ;Sets timer 1 control register
                                        ;for: Bit 12,11 = 01 continuous
                                        ;up down count mode, Bit 6 = 1
                                        ;enables timer, Bit 1 = 1
                                        ;enables timer compare operation
        SPLK    #1000100001000000b, T2CON
                                        ;Sets timer 2 control register
                                        ;for: Bit 12,11=01 continuous up
                                        ;down count mode, Bit 6 = 1
                                        ;enables timer
        SPLK    #0000000000000001b, EVAIMRB
                                        ;Enables interrupt upon timer 2
                                        ;period occurrence
        SPLK    #0000000000000000b, EVAIFRB
                                        ;Reset corresponding interrupt
                                        ;flags
        LDP     #0h                     ;Set data page corresponding to
                                        ;registers
        SPLK    #0000000000000100b, IMR
                                        ;Enable level 3 (INT3)
                                        ;interrupts
        CLRC    INTM                    ;Enable interrupts
LOOP:   B       LOOP                    ;Program infinitely loops here
                                        ;while waiting for interrupts
```

7.8 Interrupt Service Routine

Once an interrupt has occurred, the algorithm will perform several tasks as shown in Fig. 7.12. After the sensor voltage and current values are obtained, the algorithm either returns to the main wait loop or branches to the regulation code.

The code sequence below is in charge of identifying what event caused the interrupt. Reading the PIVR register obtains contains the identification number of the occurring interrupt. If the PIVR number corresponds to the Timer 2 period match interrupt (#002Bh), then the DSP branches to the regulation code. Otherwise, it branches back to the wait loop (LOOP:) in the main code.

```
PERIOD:LDP     #PIVR>>7h            ;Set data page corresponding to
                                    ;PIVR register
       LACL    PIVR                 ;Load content of PIVR register
                                    ;to accumulator
       SUB     #002Bh               ;Subtract number of timer 2
                                    ;period match interrupt
       BCND    REGULATION, EQ       ;If content matches, then branch
                                    ;to regulation code
       CLRC    INTM                 ;Otherwise clear interrupt mask
                                    ;to re-enable interrupts
       RET                          ;And return to wait loop in main
                                    ;code
```

7.8.1 Reading Voltage Sensors

This section of code contains protection against negative values that may occur because of physical sensor drift. A negative value must be eliminated. The probe offset is determined manually when physically connecting the DSP to the converter. The block of code below reads in the voltage from the ADC result register.

```
       LDP     #RESULT0>>7h         ;Set data page corresponding to
                                    ;ADC registers
       LACL    RESULT0              ;Load result register 1 content
                                    ;(i.e. current sample) to
                                    ;accumulator
       LDP     #06h                 ;Set data page corresponding to
                                    ;variables
       SUB     V_OFFSET             ;Subtract voltage probe offset
       BCND    S1, GEQ              ;If the result is positive or
                                    ;zero, then branch to proceed
                                    ;normally
       LACL    #0000h               ;Otherwise, set result to zero
S1:    SACL    V_SAMPLE, 10         ;Right shift result by 10 bits
                                    ;and store it. The 6-bit shift
```

```
                                    ;is because the 6 LSBs of the
                                    ;result register are
                                    ;insignificant. The 4-bit shift
                                    ;is for formatting purposes.
```

7.8.2 Formatting the Voltage Sample

The voltage sample read from the A/D converter needs to be multiplied by the value 14d (=Eh) in order to change the value into the 16mV per digit format.

```
        LDP     #06h                ;Set data page corresponding to
                                    ;variables
        LT      V_SAMPLE            ;Load voltage sample to
                                    ;multiplier
        MPY     #00Eh               ;Multiply by 14d
        SPL     V_SAMPLE            ;Store 16 least significant bits
                                    ;of the result to V_SAMPLE
```

7.8.3 Reading the Current Sensors

This code is similar to the code that reads the voltage sensors, with the exception of the channel read, which is channel 1 here instead of channel 0. As with the voltage reading code, the code below reads the result register of the ADC that contains the result from the current measurement.

```
        LDP     #RESULT1>>7h        ;Set data page corresponding to
                                    ;ADC registers
        LACL    RESULT1             ;Load result register 1 content
                                    ;(i.e. current sample) to
                                    ;accumulator
        LDP     #06h                ;Set data page corresponding to
                                    ;variables
        SUB     I_OFFSET            ;Subtract current probe offset
        BCND    S2, GEQ             ;If result is positive or zero
                                    ;then branch to proceed normally
        LACL    #0000h              ;Otherwise, set result to zero
S2:     SACL    I_SAMPLE, 6         ;Right shift result by 6 bits
                                    ;and store it. Right shift is
                                    ;because the 6 LSBs are
                                    ;insignificant
```

7.8.4 Filtering the ADC Readings

This code accumulates the voltage and current samples from every interrupt in order to calculate their averages once every PWM cycle.

```
        LDP     #06h                ;Set data page corresponding to
```

```
                                ;variables
LACL    V_SUM                   ;Load voltage sample sum to
                                ;accumulator
ADD     V_SAMPLE                ;Add voltage sample to
                                ;accumulator
SACL    V_SUM                   ;Store result as voltage sum
LACL    I_SUM                   ;Load current sample sum to
                                ;accumulator
ADD     I_SAMPLE                ;Add current sample to
                                ;accumulator
SACL    I_SUM                   ;Store result as current sum
```

7.9 The Regulation Code Sequences

The following sequences described in this section execute once every Timer 2 period interrupt. The Timer 2 period interrupt is set to occur at a frequency of 16 times that of the PWM switching frequency. This ensures that the regulation is calculated only once every PWM cycle.

The code checks the counter of interrupt occurrence (INT_CNT). Every time the four least significant bits are equal to 15 (every 16 interruptions), the DSP branches to the regulation code. Otherwise, it returns directly from the interrupt service routine.

```
LDP     #06h                    ;Set data page corresponding to
                                ;variables
LACL    INT_CNT                 ;Load interrupt occurrence
                                ;counter into accumulator
ADD     #1h                     ;Increment by 1
SACL    INT_CNT                 ;Store back as interrupt
                                ;occurrence counter
AND     #000Fh                  ;Discard all bits but 4 LSBs
XOR     #000Fh                  ;Check for equality with 16d=Fh
BCND    S_RET, NEQ              ;If not, then branch to return
                                ;from interrupt
```

7.9.1 Calculating the Voltage and Current Average Values

This sequence takes the sum of 16 voltage and current samples and divides it by 16. The division is performed with a 4-bit right shift. The results are the averaged values of the load voltage and current.

```
LDP     #06h                    ;Set data page corresponding to
                                ;variables
LACL    V_SUM                   ;Load sum of voltage sample to
                                ;accumulator
SACL    V_AVERAGED, 4           ;Shift right by 4 bits and store
```

```
                                ;as the average load voltage
SPLK    #0000h, V_SUM           ;Reset sum of voltage samples
LACL    I_SUM                   ;Load sum of current samples to
                                ;accumulator
SACL    I_AVERAGED, 4           ;Shift right by 4 bits and store
                                ;as the average load current
SPLK    #0000h, I_SUM           ;Reset sum of current samples
```

7.9.2 Voltage Comparator

This code simply outputs the difference between the voltage reference (V_REF) and the averaged load voltage (V_AVERAGED) as the error (V_ERROR).

```
LDP     #06h                    ;Set data page corresponding to
                                ;variables
ACL     V_REF                   ;Load voltage reference in the
                                ;accumulator
SUB     V_AVERAGED              ;Subtract the averaged load
                                ;voltage. The accumulator now
                                ;contains the difference
SACL    V_ERROR                 ;Store the result as V_ERROR
```

7.9.3 Current Regulation

This sequence checks the averaged load current versus the predefined limit and, if necessary, stops the integration of the voltage error and decrements the integrator.

```
LDP     #06h                    ;Set data page corresponding to
                                ;variables
LACL    I_LIMIT                 ;Load maximum current value to
                                ;accumulator
SUB     I_AVERAGED              ;Subtract actual averaged
                                ;current value
BCND    S3, GEQ                 ;If actual current below maximum
                                ;then branch to proceed normally
SPLK    #0000h, V_ERROR         ;Otherwise, stop integrating
                                ;voltage error
LACL    INTEGRAL                :And load integral value
SUB     #1h                     ;Decrement it by 1
SACL    INTEGRAL                ;Store it back as integral value
```

7.9.4 PI Regulator

The code below is actually only an integral regulator, which proved to be sufficient for the application.

```
S3:     LDP     #06h                ;Set data page corresponding to
                                    ;variables
        LACL    INTEGRAL            ;Load integral value to
                                    ;accumulator
        ADD     V_ERROR             ;Integrate the error
        SACL    INTEGRAL            ;Store result as integral value
        SACL    ACTION, 3           ;Right shift by 3 bits for
                                    ;formatting purposes and output
                                    ;result as action
```

7.9.5 Short Circuit Protection

The protection algorithm is activated if the average current rises to twice the limit. This sequence executes along with the rest of the regulation code. The protection algorithm must immediately reset the PWM output and the integrator to zero in order to protect the buck-boost converter and load against sudden surges of current. It should be noted that in case of a short-circuit, this protection will be activated only a few times. Once the filter capacitor is discharged, the load current will drop to acceptable levels and the LF2407 will work in current regulation mode.

```
        LDP     #06h                ;Set data page corresponding to
                                    ;variables
        LACL    I_LIMIT             ;Load current limit to
                                    ;accumulator
        SFL                         ;Multiply by 2
SUB     I_AVERAGED                  ;Subtract the average load
                                    ;current value
        BCND    S_PWM, GEQ          ;If the average load current is
                                    ;below the critical limit, then
                                    ;proceed normally to outputting
                                    ;the action to PWM
        SPLK    #0000h, ACTION      ;Otherwise, reset the PWM
                                    ;output through the ACTION
                                    ;signal
        SPLK    #0000h, INTEGRAL    ;And reset the PI regulator
                                    ;integrator. The proceed to
                                    ;outputting the action to the
                                    ;PWM hardware
```

7.9.6 Output Action to PWM

The action signal from the PI regulator should be linearized (i.e., transformed into a corresponding duty cycle) for optimum dynamic performance. However, dynamic performance is ensured by the processing speed of the DSP. In addition, the actual relationship between the duty cycle and the voltage gain is dependent

upon the load and converter characteristics and would therefore have to be identified for each specific application. The duty cycle effectively output by the PWM hardware as it is programmed here is the following function:

$$d = \frac{T1PR - T1CMPR}{T1PR} \tag{7.5}$$

Therefore, the code must adapt the action signal from the PI regulator (ACTION) into a value to be stored in the T1CMPR register. The equation used is

$$\frac{ACTION}{T1PR} = \frac{T1PR - T1CMPR}{T1PR} \tag{7.6}$$

which yields the simple transformation implemented in the code sequence below.

$$T1CMPR = T1PR - ACTION \tag{7.7}$$

```
        LDP     #GPTCONA>>7h          ;Set data page corresponding to
                                      ;PWM timer registers
        LACL    T1PR                  ;Load period timer value
        LDP     #06h                  ;Set data page corresponding to
                                      ;variables
        SUB     ACTION                ;Subtract the action signal from
                                      ;the PI regulator. The result is
                                      ;the PWM timer compare value
        LDP     #GPTCONA>>7h          ;Set data page corresponding to
                                      ;the PWM timer registers
        SACL    T1CMPR                ;Store calculated compare value.
                                      ;It is now effectively the value
                                      ;that will be used by the PWM
                                      ;hardware for the next PWM cycle
```

7.9.7 Return to Main Code

This code clears the flag corresponding to the Timer 2 period match interrupt in EVAIFRB and re-enables the interrupts. It branches back to where the program was before the interrupt, i.e., the wait loop in the main code.

```
S_RET:  LDP     #EVAIFRB>>7h          ;Set data page corresponding to
                                      ;event manager A registers
        LACL    EVAIFRB               ;Load value of event manager A
                                      ;interrupt flags register
        SACL    EVAIFRB               ;Store it back to the register.
                                      ;This effectively clears the
                                      ;interrupt flags
        CLRC    INTM                  ;Re-enable the interrupts
        RET                           ;Return from interrupt
```

7.10 Results

The following waveforms were captured from a physical buck-boost converter under the control of the DSP algorithm described in this chapter. The waveform of the diode current is shown first in Fig. 7.13.

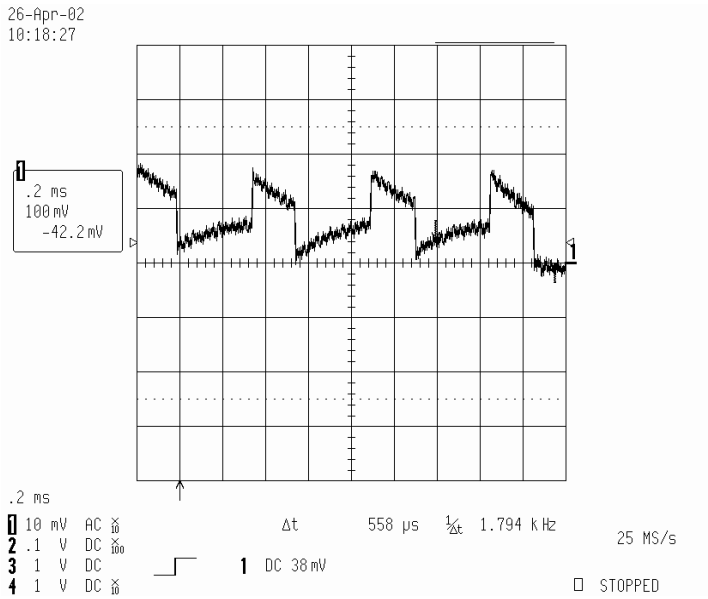

Figure 7.13 Diode current ripple.

The current waveform displays a ripple that emphasizes the need for an averaging filter. The ripple observed has an amplitude of about 15mA, with an average value of 17mA. The waveform also displays high frequency noise, which further reduces the precision of the measurements, requiring the use of a filter.

The voltage waveform in Fig. 7.14 shows the ripple due to the charge of the capacitor during the switch-off period and the discharge of the capacitor by the load during the switch-on period. The amplitude of the ripple is only 10mV, but its high frequency makes an output filter mandatory. Furthermore, the load used in these waveforms is a low current load. A larger load would make the ripple much more significant, thus also requiring an averaging filter.

The controlled buck-boost converter was tested in normal load conditions, with an increased load, and in sudden short circuit. It held the voltage at the predetermined reference value (5V) under normal load conditions (300Ω). A current limit was set at 25mA. When the load was decreased to 150Ω, the control entered in current regulation mode. The current was held precisely at 25mA, resulting in a voltage of 3.76V. When the output was suddenly short-circuited, the converter reacted immediately and held the current to 28mA instead of 25mA. The reason for this is that the output voltage was only a few mV, which is very close to the

3.224mV of the ADC. It is impossible for the controller to see the difference between 25 and 28mA, and hence impossible to regulate the current to exactly 25mA.

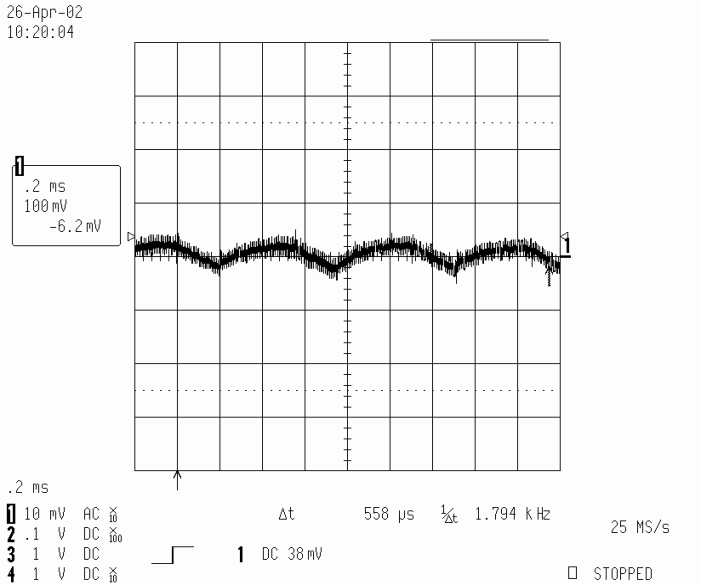

Figure 7.14 Load voltage ripple.