

Application Note

**DC Brushed Motor Control
with the 78K0S
Low Pincount Microcontroller**

NOTES FOR CMOS DEVICES

① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

Note:

Strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it once, when it has occurred. Environmental control must be adequate. When it is dry, humidifier should be used. It is recommended to avoid using insulators that easily build static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work bench and floor should be grounded. The operator should be grounded using wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with semiconductor devices on it.

② HANDLING OF UNUSED INPUT PINS FOR CMOS

Note:

No connection for CMOS device inputs can be cause of malfunction. If no connection is provided to the input pins, it is possible that an internal input level may be generated due to noise, etc., hence causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using a pull-up or pull-down circuitry. Each unused pin should be connected to V_{DD} or GND with a resistor, if it is considered to have a possibility of being an output pin. All handling related to the unused pins must be judged device by device and related specifications governing the devices.

③ STATUS BEFORE INITIALIZATION OF MOS DEVICES

Note:

Power-on does not necessarily define initial status of MOS device. Production process of MOS does not define the initial operation status of the device. Immediately after the power source is turned ON, the devices with reset function have not yet been initialized. Hence, power-on does not guarantee out-pin levels, I/O settings or contents of registers. Device is not initialized until the reset signal is received. Reset operation must be executed immediately after power-on for devices having reset function.

All other product, brand, or trade names used in this publication are the trademarks or registered trademarks of their respective trademark owners.

Product specifications are subject to change without notice. To ensure that you have the latest product data, please contact your local NEC Electronics sales office.

- **The information in this document is current as of July, 2005. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".

The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.

"Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots.

"Special": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support).

"Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact an NEC Electronics sales representative in advance to determine NEC Electronics' willingness to support a given application.

(Note)

- (1) "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.
- (2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

M8E 02.11-1

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics America Inc.

Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC Electronics (Europe) GmbH

Duesseldorf, Germany
Tel: 0211-65 03 1101
Fax: 0211-65 03 1327

Sucursal en España

Madrid, Spain
Tel: 091- 504 27 87
Fax: 091- 504 28 60

Succursale Française

Vélizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

Filiale Italiana

Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

Branch The Netherlands

Eindhoven, The Netherlands
Tel: 040-244 58 45
Fax: 040-244 45 80

Branch Sweden

Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

United Kingdom Branch

Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Hong Kong Ltd.

Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.

Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.

Singapore
Tel: 65-6253-8311
Fax: 65-6250-3583

NEC Electronics Taiwan Ltd.

Taipei, Taiwan
Tel: 02-2719-2377
Fax: 02-2719-5951

Table of Contents

Chapter 1	Introduction.....	9
Chapter 2	DC Motor Description.....	10
Chapter 3	H-Bridge Description.....	12
Chapter 4	PWM Description.....	13
Chapter 5	NEC 78K0S Microcontroller – μPD78F9222 Description.....	14
Chapter 6	Implementing DC Motor Control.....	15
Chapter 7	Possibilities and Conclusion.....	20
Chapter 8	Software Listing.....	21

List of Figures

Figure 2-1:	DC Brushed Motor.....	10
Figure 2-2:	Motor Operation.....	11
Figure 3-1:	The H-bridge.....	12
Figure 4-1:	PWM Duty Cycle	13
Figure 5-1:	μ PD78F9222 Microcontroller.....	14
Figure 6-1:	Software PWM.....	16
Figure 6-2:	Circuit Arrangement.....	17
Figure 6-3:	Flowchart (main program)	18
Figure 6-4:	Flowchart (interrupt service routine).....	19

List of Tables

Table 6-1: μ PD78F9222 Timers	15
---	----

Chapter 1 Introduction

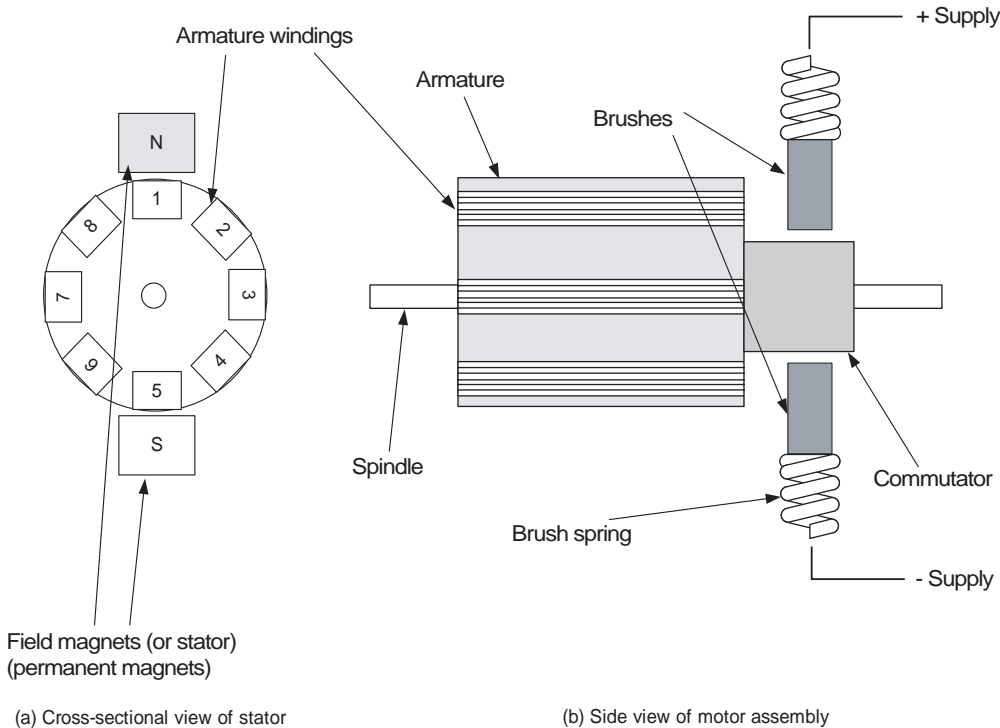
An electric motor converts electrical energy into rotational force. They are commonplace within household goods (e.g. hair dryer, electric shaver), toys, white goods, automotive systems (engine fan, air conditioning fan) and building management systems (HVAC, CCTV systems). In some simple systems the motor will need to rotate in one direction and at a fixed speed only. On others, bidirectional control will be needed at variable speed, which is the subject of this application note which describes a low cost DC motor control application using NEC 78K0S Low Pincount microcontrollers.

It covers the issues involved in driving and controlling a DC motor and is intended to help users understand the relevant peripherals of the Low Pincount devices.

The published software and hardware configurations are meant to serve as examples only and are not intended for mass production.

Chapter 2 DC Motor Description

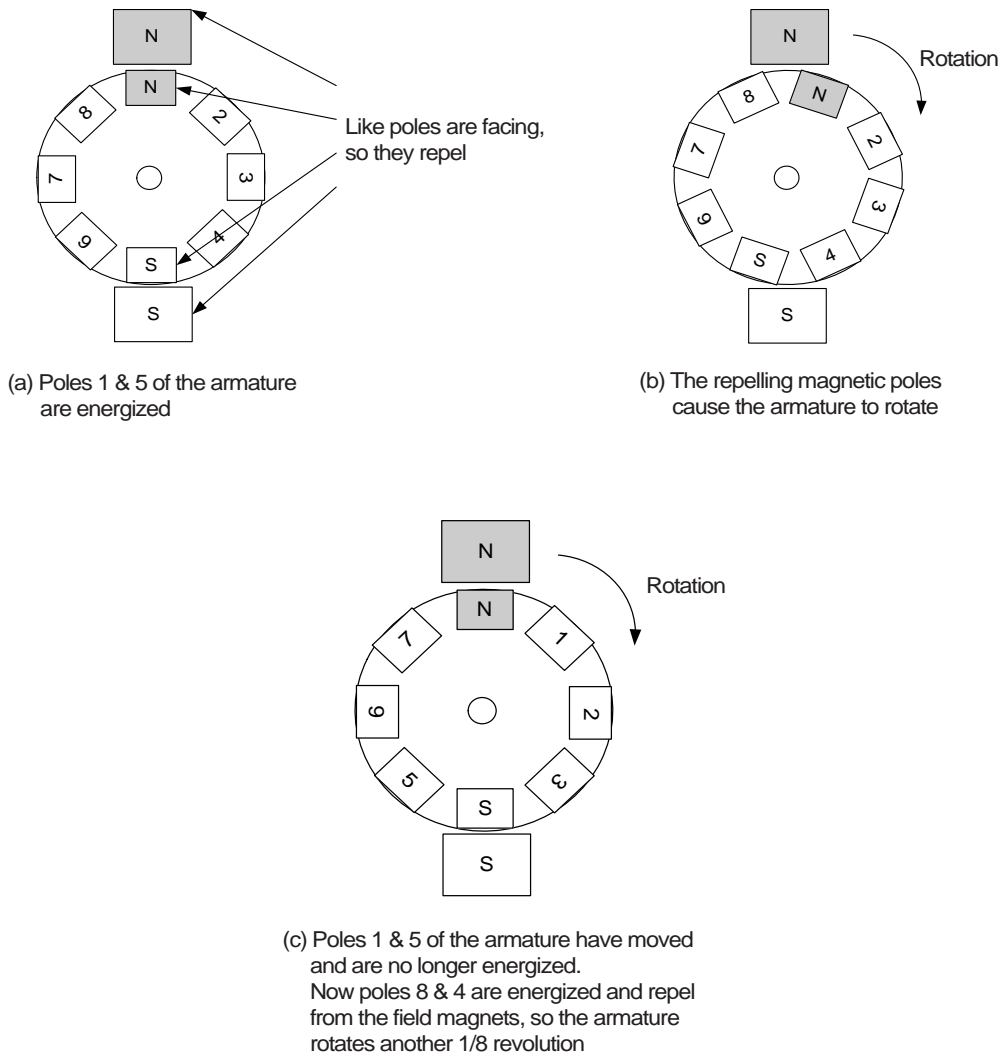
Figure 2-1: DC Brushed Motor



A typical DC brushed motor consists of a fixed part (the stator) and a moving part (the armature). The stator consists of a number of fixed permanent magnets of alternating polarity (see Figure 2-1a). The armature is mounted on a rotating spindle, and consists of coils of wire formed around a core. Each end of these wires is connected to a copper contact, also mounted on the spindle, called the commutator. The purpose of the commutator is to provide electrical contact with the brushes, sprung contacts between which the commutator revolves. As the armature (and commutator) turn, electrical power is connected to each of the armature windings in turn.

Figure 2-2 shows the operation of the motor.

Figure 2-2: Motor Operation



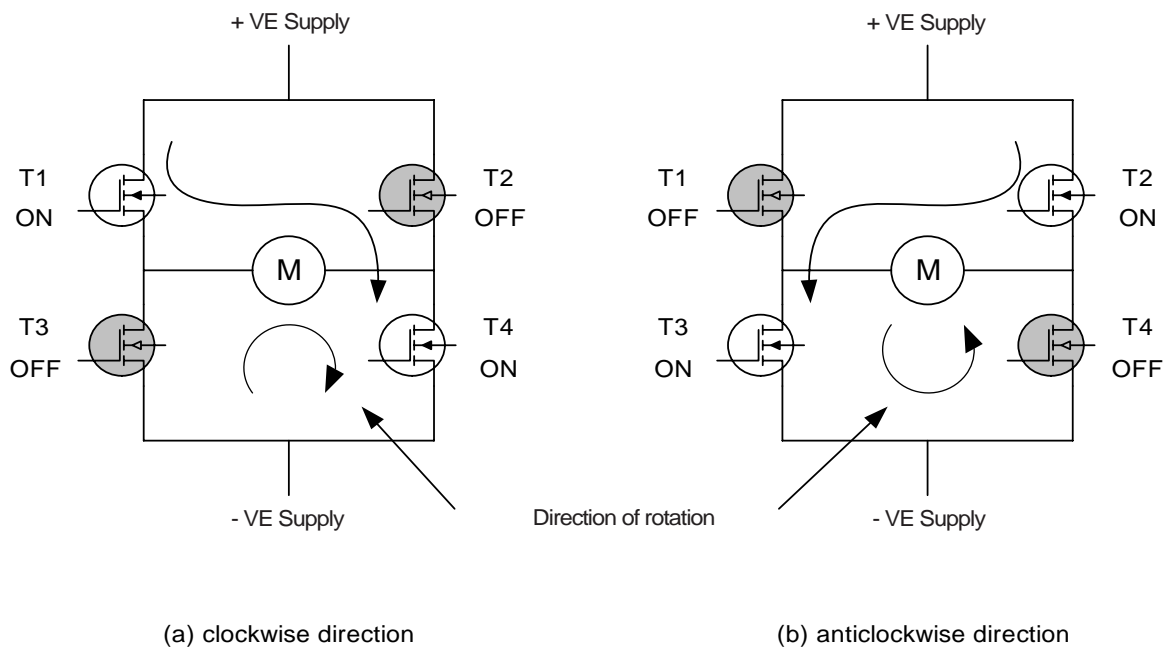
Consider the motor as it is in Figure 2-2 (a). DC current flows through the brushes and commutator and through armature coil 1-5. This coil, together with the armature material (e.g. iron) produces a magnetic field that will repel from the poles of the stator (like poles attract, opposite poles repel). This effect will cause the rotor to turn, as in Figure 2-2 (b), so armature coil 4-8 is now beginning to align with the poles of the stator. However, due to the action of the commutator, at this moment the electrical supply to coil 1-5 is broken, and now applied to coil 4-8 as in Figure 2-2 (c). Again, the coil is producing a magnetic field which opposes that of the field magnets and the repelling force causes the armature to rotate through another step.

This process repeats indefinitely and produces the rotary motion of the armature. The momentum of the rotating armature reduces the effect of the discrete “steps” that are made by the motor, as does increasing the number of armature coils.

The speed and torque of the motor depend upon the strength of the magnetic field generated by the energized windings of the motor, which depend on the current through them. Therefore adjusting the armature voltage (and current) will change the motor speed. In this application note speed control is based on generating and varying a PWM signal from the microcontroller.

Chapter 3 H-Bridge Description

Figure 3-1: The H-bridge



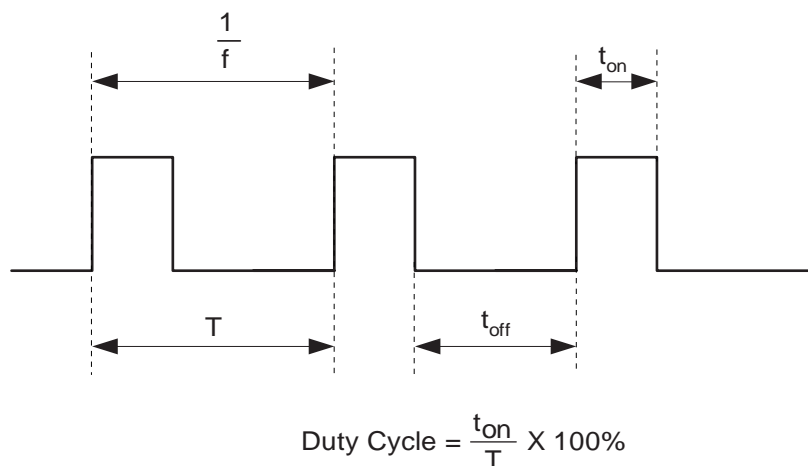
The circuit of Figure 3-1 shows a common method of driving a DC motor in both directions. It can be seen that to change direction of the motor the direction of the current through it must be reversed. This is achieved with the H-bridge. In Figure 3-1 (a) N-channel MOSFETs T1 and T4 are switched on, and T2 and T3 are off, resulting in the current direction (and motor direction) shown. To reverse the direction of the motor the opposite pair of transistors are turned off and on (Figure 3-1 (b)). T1 and T2 are known as *high side* drivers whilst T3 and T4 are *low side* drivers.

It should be noted that while the gates of the low side drivers may be driven directly by CMOS logic outputs, the high side drivers need additional MOSFET driver circuitry, since to turn on the transistor V_{GS} must be positive. Full explanation of MOSFET driving is beyond the scope of this application note.

Chapter 4 PWM Description

PWM (Pulse Width Modulation) relies upon a fixed (high) frequency pulse waveform of variable duty cycle (see Figure 4-1).

Figure 4-1: PWM Duty Cycle

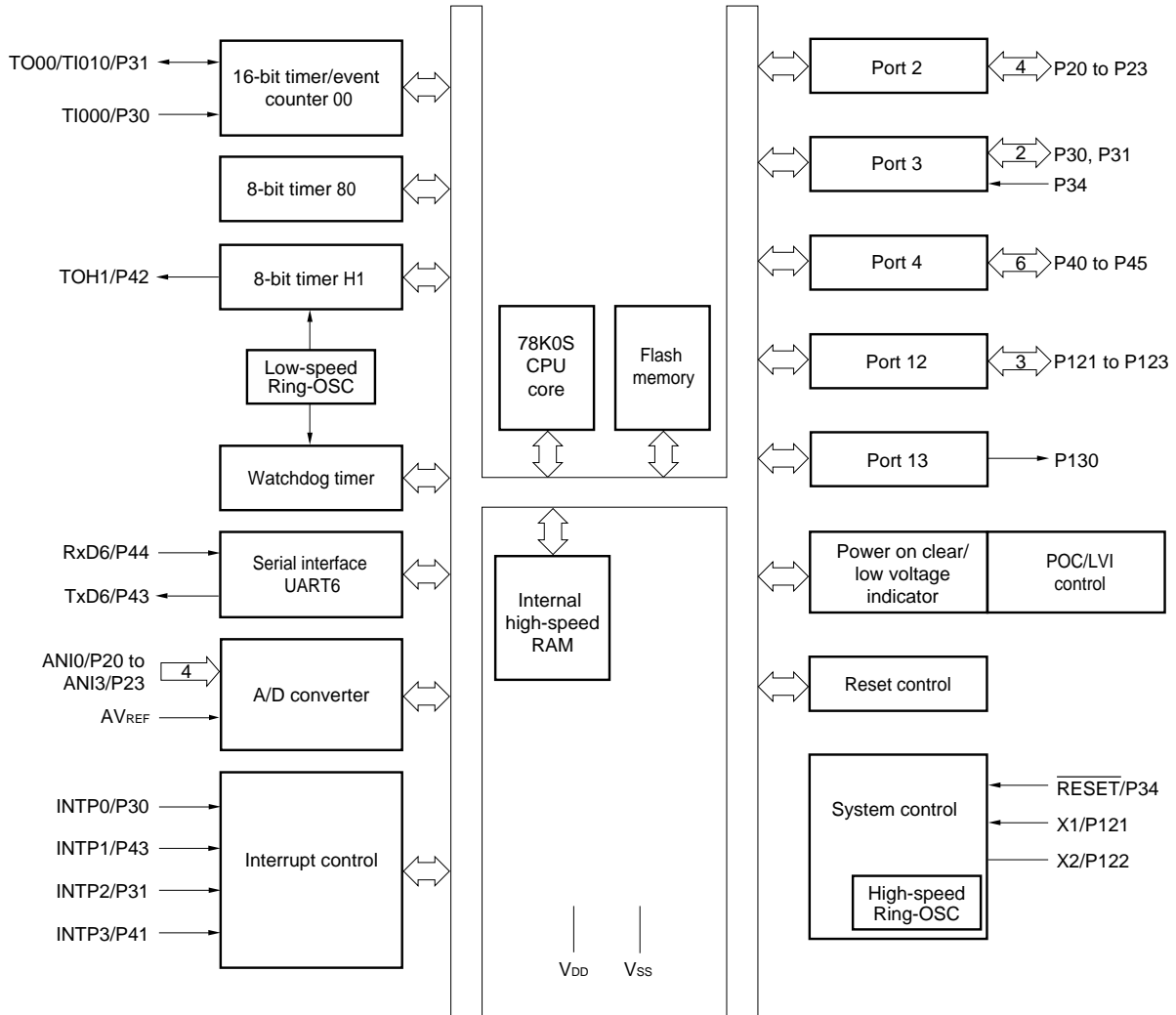


The duty cycle is defined as the ratio of the “on” time to the total period of the pulse. So, for example, if the period is 100 μS and the pulse width is 50 μS , the duty cycle is 50%. The significance of this is that if the PWM signal is averaged, either electrically or, in the case of a motor, mechanically, the resulting signal will be DC and have 50% of its original amplitude. This enables effective generation of a variable DC voltage from a single digital I/O port on a microcontroller. If the MCU is switching a voltage of, say, 12 V with duty cycle of 0% to 100% variable in 256 steps, then a 0-12 V DC generator than can be set in steps of $12/256 = 0.047$ V will result.

Chapter 5 NEC 78K0S Microcontroller – μ PD78F9222 Description

The μ PD78F9222 is a 20-pin microcontroller based upon NEC's proven 78K0S core. A block diagram is shown below.

Figure 5-1: μ PD78F9222 Microcontroller



As well as the 78K0S core, the device features 4 kBytes Flash memory, 256 bytes RAM, general purpose input / output (GPIO), timers, analogue to digital converter and UART, as well as system supervisory functions such as power-on reset, low voltage indicator and watchdog timer. In addition a self-contained high speed oscillator (8 MHz) is included, allowing the device to operate with no external parts.

Chapter 6 Implementing DC Motor Control

As was mentioned previously, pulse width modulation (PWM) can be used to generate a variable analogue voltage from a digital I/O pin. If such a PWM signal is applied to the gate of T1 or T2 in Figure 3 instead of a fixed DC voltage, this will allow variable speed control of the motor. The μ PD78F9222 MCU has three timers, two of which can generate PWM signals. See Table 1 below.

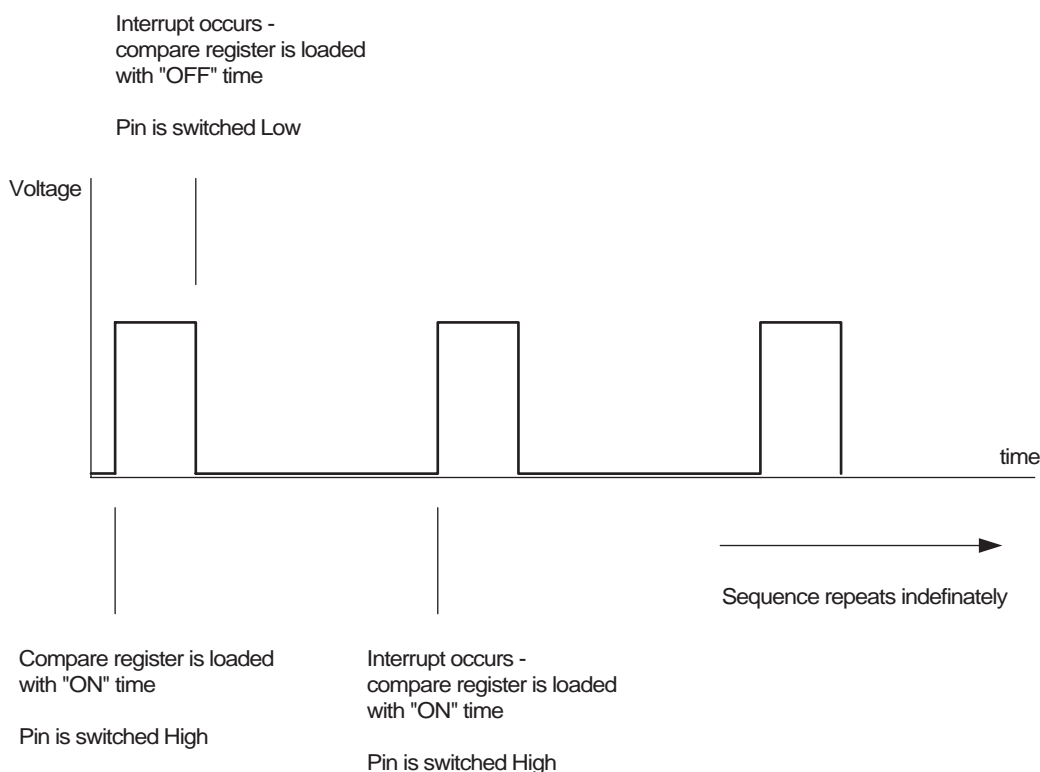
Table 6-1: μ PD78F9222 Timers

Timer 00 (16-bit)	Timer H1 (8-bit)	Timer 80 (8-bit)
Has additional functions (event counter, one-shot pulse generation, pulse width measurement). PPG (Programmable Pulse Generator) can generate PWM waveforms but must be used with care – counter must be stopped during reload to avoid glitches	Can generate PWM (8 bit)	No PWM facility

From Table 6-1 it is seen that timer 00 can be used for PWM, as can Timer H1. Whilst a motor control system could easily be realised with these timers, it was thought best to avoid the use of timer 00, since as it has the highest resolution and the most flexibility it may be useful (or required) for another part of the system. However this leaves just one 8 bit timer (H1) with PWM functionality and two are needed for bidirectional control. Only one PWM signal is needed at any time, however, and if it could be routed to more than one I/O pin under software control it would be ideal. Unfortunately this is not possible; external logic (and therefore additional components) would be needed to switch the PWM output from the MCU to the gate of either T1 or T2.

What has been implemented here is software based PWM. Instead of using the timer's hardware to produce a PWM output directly, the compare feature of timer H1 is used. A more detailed description follows:

Figure 6-1: Software PWM



With the device running at 5 MHz, timer H1 is set to be clocked at $f_{\text{clk}}/4 = 1.25$ MHz. For 8-bit PWM (256 increments) the PWM frequency will therefore be $1.25 \text{ MHz}/256 = 4.88$ kHz.

The PWM output is switched high, the H1 timer compare register is loaded with the “on” time (0-255), and the “off” time is calculated as $(256 - \text{“on” time})$. An interrupt occurs when the timer value equals the compare register value, and the output is switched low and the compare register loaded with the “off” time. The advantage of doing this is that the PWM output is switched by software and not by PWM module hardware, so under software control the PWM signal may appear at any I/O pin – thus allowing bidirectional control of the motor. The disadvantage is that PWM duty cycles close to 0% and 100% are not possible due to the time taken by the MCU to respond to the interrupt and reload the compare register (in addition to any other tasks it may be undertaking).

Figure 6-3 shows the flowchart for the code used to demonstrate the software PWM principle. A voltage between V_{DD} and 0 V is applied to analogue input ANI0. This sets the motor speed. As was previously mentioned, this method of generating PWM does not permit duty cycles at either extreme of the range – the code listing shows the ADC reading is limited to values between 20 and 240. Different arrangements and different clock speeds will allow these values to be changed as necessary by the user. P30 is set as an input to read the direction switch position (pull up to V_{DD} is enabled and input is denounced by software). Notice that MOSFET drivers may be active low or active high – for this prototype the low side drivers were active high and the high side drivers active low. This is shown in the code listing, with code for the other options commented out. It is important to enable the correct lines of code to avoid damage to the MOSFETs. *The flowchart assumes all MOSFET drivers are active high.* Notice also that when the direction changes, delays operate between switching over of the PWM outputs and the low side drivers. This is to avoid *shoot-through*, which is a large current surge caused by both high side and low side MOSFETs being on simultaneously, which will be destructive to the transistors.

The interrupt service routine (ISR) (Figure 6-4) simply checks for the current motor direction and if the appropriate PWM output is low it will be switched high and the correct PWM high time loaded into the compare register. If high it will be switched low and the correct PWM low time loaded. Two down-

counters (adc_timer and dir_timer) decrement upon every interrupt and are used to initiate an ADC conversion and perform a switch read periodically. They are reloaded with the values ADC_TIMER_RELOAD_VAL and DIR_TIMER_RELOAD_VAL respectively. This keeps the use of the hardware timers to a minimum.

The choice of MOSFET and driver used is beyond the scope of this application note – they are dependant on motor voltage and current and other factors, but the general circuit arrangement is shown below in Figure 6-2.

Figure 6-2: Circuit Arrangement

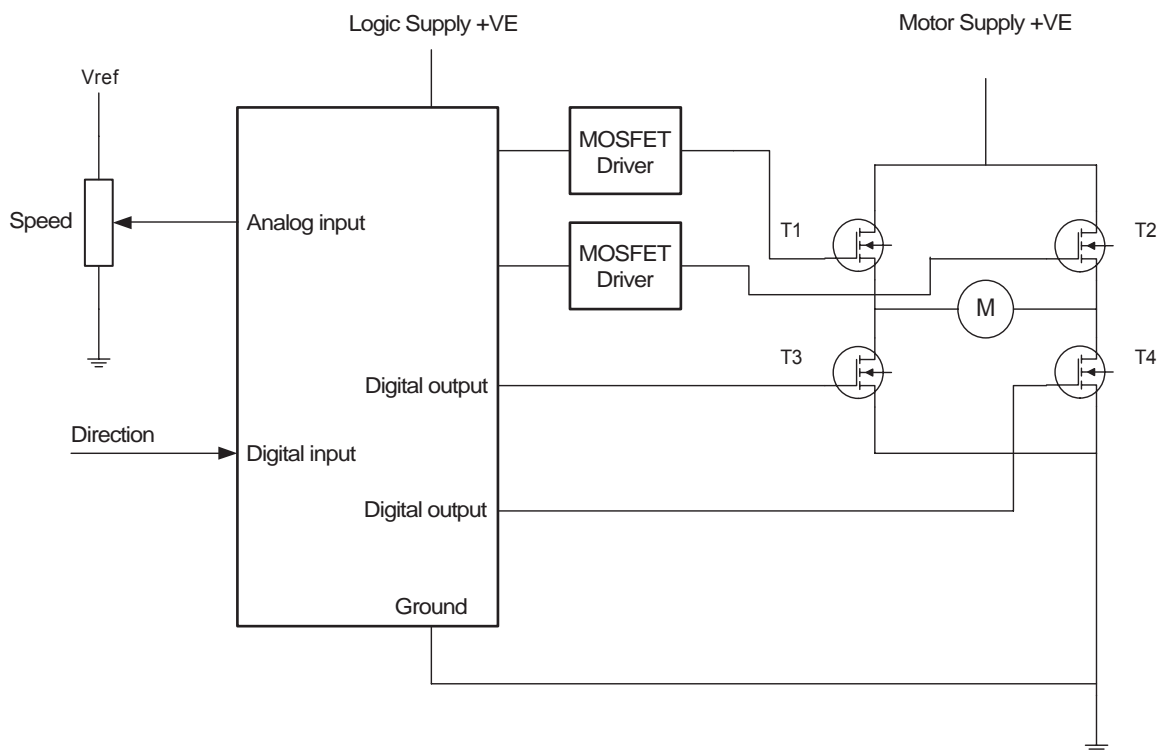


Figure 6-3: Flowchart (main program)

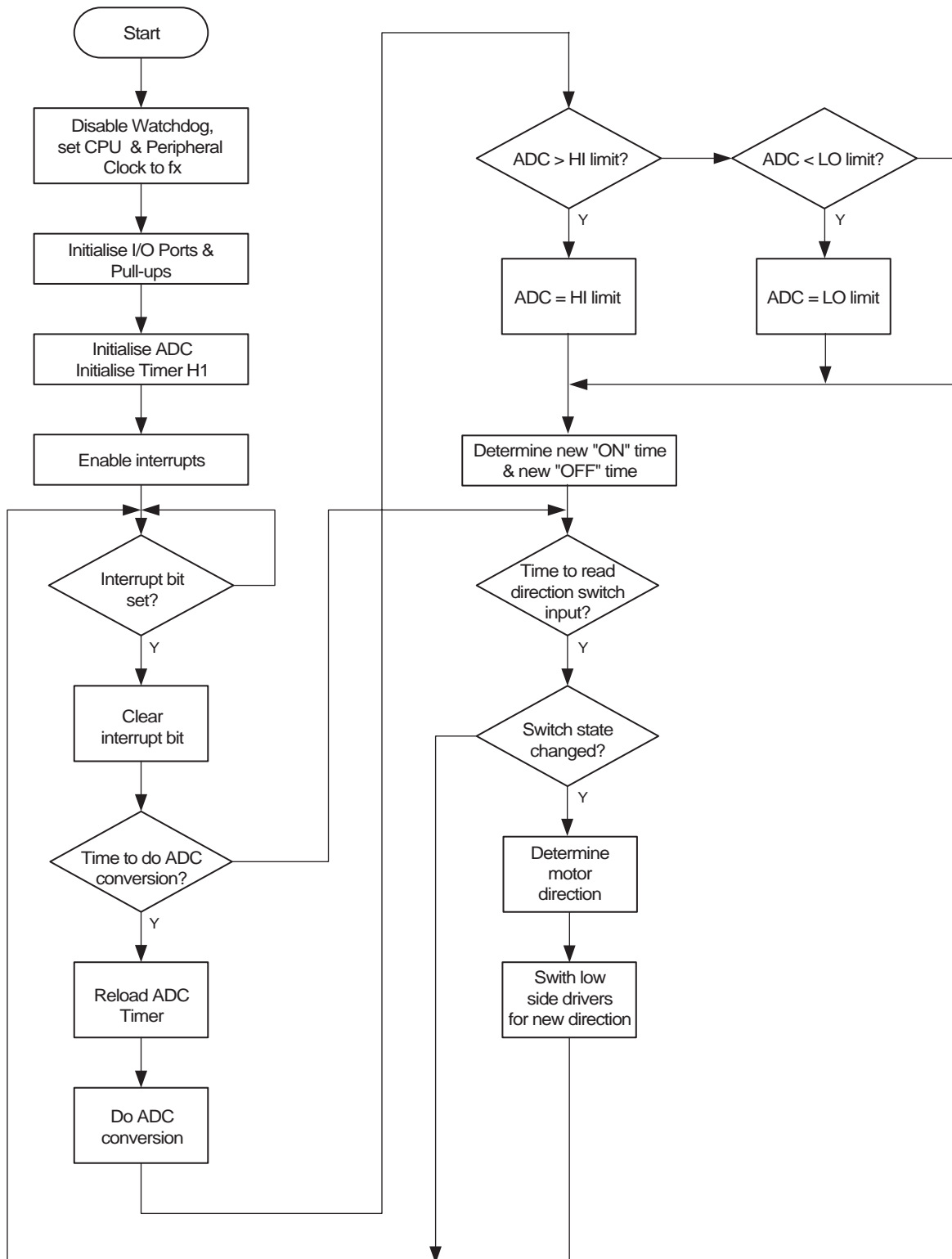
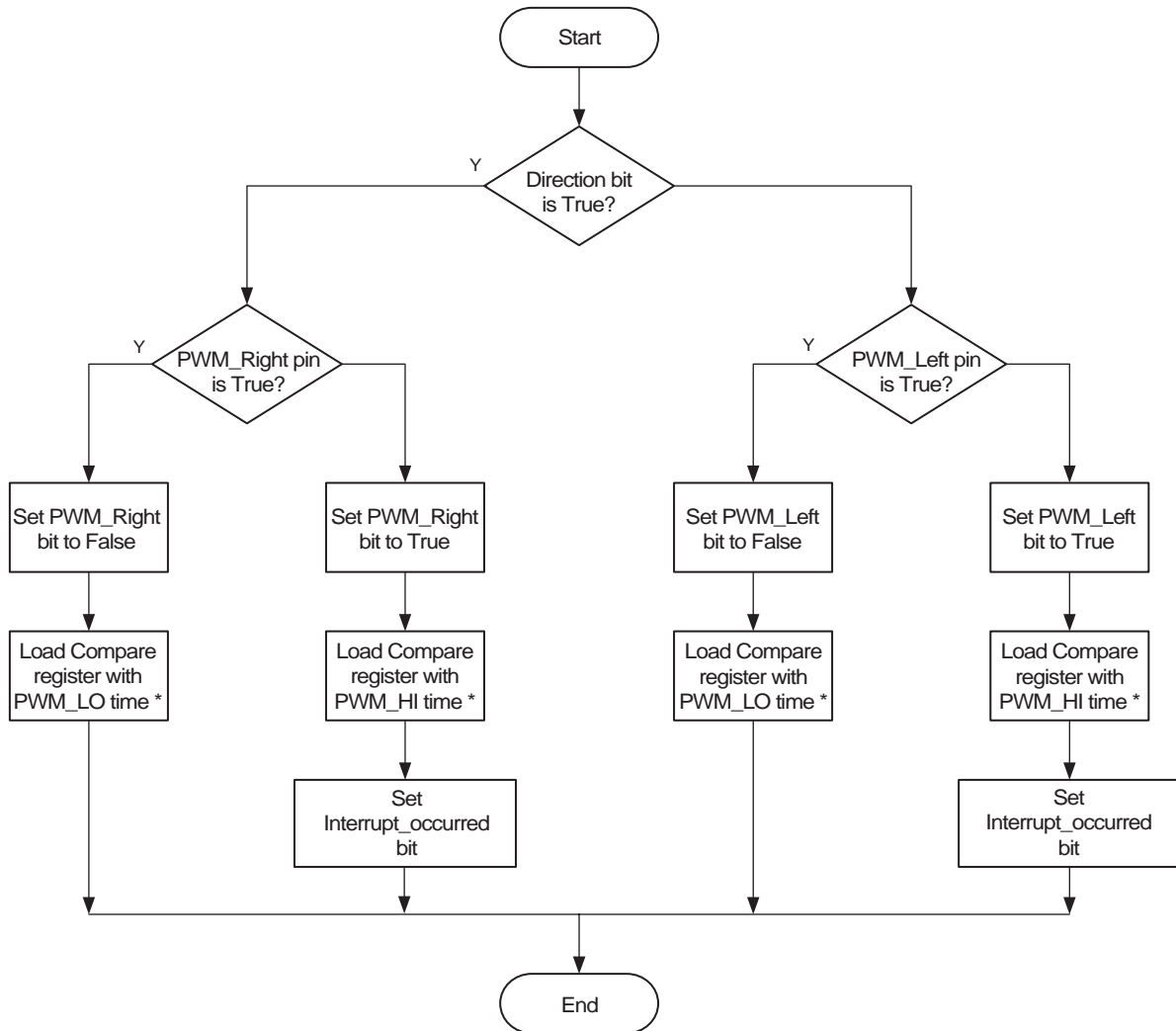


Figure 6-4: Flowchart (interrupt service routine)



Remark: * - see text

Chapter 7 Possibilities and Conclusion

This application note has shown how bidirectional control of a DC brushed motor is possible with NEC's low Pincount microcontrollers, using just one timer to generate a PWM signal that is "switchable" onto any I/O pin. There is no reason this idea could not be expanded to drive more than one motor, still keeping as many of the MCU's resources as possible free.

The μ PD78F9222 device featured here has a LIN enabled UART making it useful for automotive applications – the idea presented here could be adapted to a full-featured LIN slave node controlling, for example, door mirror motors. HVAC and other building management systems could benefit from such low cost local motor control.

The code in this project was built with IAR Embedded Workbench V4.3 and tested with NEC's 78K0S Minicube In-Circuit Emulator.

Memory used (with stack set to 32 bytes):

- 476 bytes of CODE memory
- 39 bytes of DATA memory (32 for stack, 7 for application)

Chapter 8 Software Listing

```
#include <io78f9222.h>
#include <intrinsics.h>

#pragma location = "OPTBYTE"
__root const unsigned char optbyte = 0x92;    // Shortest stabilisation time 2^10/fx
                                              // RESET pin is used as is
                                              // External clock input
                                              // shortest stabilisation time 2^10/fx
                                              // Ring osc can be stopped

#define TRUE 1
#define FALSE 0
#define INPUT 1
#define OUTPUT 0

#define PWM_LEFT_PIN P4_bit.no2
#define PWM_LEFT_DIR PM4_bit.no2
#define PWM_RIGHT_PIN P4_bit.no3
#define PWM_RIGHT_DIR PM4_bit.no3

#define LOW_SIDE_LEFT_PIN P4_bit.no4
#define LOW_SIDE_LEFT_DIR PM4_bit.no4
#define LOW_SIDE_RIGHT_PIN P4_bit.no5
#define LOW_SIDE_RIGHT_DIR PM4_bit.no5

#define PUSHBUTTON_PIN P3_bit.no0
#define PUSHBUTTON_PIN_DIR PM3_bit.no0

#define ADC_TIMER_RELOAD_VAL 1000
#define DIR_TIMER_RELOAD_VAL 100

__saddr union {unsigned char flags_1; __BITS8 bit_field;};

#define switch_read_state bit_field.no0
#define switch_reading_temp bit_field.no1
#define switch_reading bit_field.no2
#define old_switch bit_field.no3
#define motor_dir bit_field.no4
#define interrupt_occurred bit_field.no5

__saddr unsigned char pwm_hi_time;
__saddr unsigned char pwm_lo_time;
__saddr unsigned int adc_timer = ADC_TIMER_RELOAD_VAL;
__saddr unsigned int dir_timer = DIR_TIMER_RELOAD_VAL;

void delay (unsigned char);
```

Chapter 8 Software Listing

```
void main(void){

    motor_dir = 1;           // default direction

    WDTM = 0x77;           // disable watchdog

    PCC = 0x00;           // set CPU clock to fx
    PPCC = 0x00;         // peripheral clock = fx

// HSRCM = 0x00;         // high speed ring oscillator operates (NOT IAR V4)
LSRCM = 0x01;         // low speed ring oscillator stops

    OSTC = 0x00;         // shortest stabilisation time 2^10/fx

    PU3 = 0x01;         // pull-up on for pushbutton
    PUSHBUTTON_PIN_DIR = INPUT;

    PMC2_bit.no0=1;     // set alternate function mode
    PM2_bit.no0=1;     // set port to input mode

    ADIF = 0;           // clear interrupt request flag
    ADS = 0;

    P4 = 0x0C;         // PWM outputs HIGH (intersil high side drivers active LOW)
                        // low side outputs LOW (intersil low side drivers active HIGH)
    PM4 = 0xC3;         // H-bridge drive pins to output, all others left as input

    pwm_hi_time = 20;   // 10 is minimum 240 is maximum
    pwm_lo_time = 255 - pwm_hi_time;

    TMHMD1 = 0x10;     // PWM clock is Fx/4. interval timer mode selected. output disabled

    TMIFH1 = 0;       // clear interrupt request flag
    TMMKH1 = 0;       // enable timer80 interrupt

    __enable_interrupt(); // global interrupt enable

    CMP01 = pwm_hi_time;

    TMHMD1 |= 0x80;    // enable timer H1

    for (;;) {

        if (interrupt_occurred != FALSE) {

            interrupt_occurred = FALSE;

            if (--adc_timer == 0) {
                adc_timer = ADC_TIMER_RELOAD_VAL;

                ADCS = TRUE; // start conversion
                while (!ADIF) // wait for EOC
                    ;
                if (ADCRH <= 20)
                    pwm_hi_time = 20;
                else if (ADCRH >= 240)
                    pwm_hi_time = 240;
                else
                    pwm_hi_time = ADCRH;

                pwm_lo_time = 255 - pwm_hi_time;

                ADCS = FALSE; // to stop continuous conversions
                ADIF = FALSE; // clear interrupt flag
            }
        }
    }
}
```

Chapter 8 Software Listing

```
if (--dir_timer == 0){
    dir_timer = DIR_TIMER_RELOAD_VAL;

    switch (switch_read_state){

        case (0):{
            switch_reading_temp = PUSHBUTTON_PIN;
            switch_read_state = 1;
            break;

        }

        case (1):{
            if (PUSHBUTTON_PIN == switch_reading_temp){ // switch denounce
                switch_reading = switch_reading_temp;

                if (switch_reading != old_switch){ // then change has occurred
                    old_switch = switch_reading;
                    if (switch_reading != 0){
                        __disable_interrupt();
                        motor_dir = 1;
                        PWM_LEFT_PIN = FALSE; // stop PWM (active high PWM driver)
                        PWM_LEFT_PIN = TRUE; // stop PWM (active low PWM driver)
                        delay (100);
                        LOW_SIDE_RIGHT_PIN = FALSE;
                        delay (100);
                        LOW_SIDE_LEFT_PIN = TRUE;
                        delay (100);
                        __enable_interrupt();
                    }
                    else{
                        __disable_interrupt();// dir = 0
                        motor_dir = 0;
                        PWM_RIGHT_PIN = FALSE; // stop PWM (active high PWM driver)
                        PWM_RIGHT_PIN = TRUE; // stop PWM (active low PWM driver)
                        delay (100);
                        LOW_SIDE_RIGHT_PIN = TRUE;
                        delay (100);
                        LOW_SIDE_LEFT_PIN = FALSE;
                        delay (100);
                        __enable_interrupt();
                    }
                }
            }
        }

        switch_read_state = 0;

        break;

    }

}

}

}

}

}
```

Chapter 8 Software Listing

```
// *****
#pragma vector = INTTMH1_vect

__interrupt void isr_INTTMH1(void){

    if (motor_dir == 1){
        if (PWM_RIGHT_PIN == TRUE){
            PWM_RIGHT_PIN = FALSE;
//            CMP01 = pwm_lo_time;// for active high PWM driver
            CMP01 = pwm_hi_time;// for active low PWM driver

        }
        else{
//            PWM_RIGHT_PIN = TRUE;
            CMP01 = pwm_hi_time;// for active high PWM driver
            CMP01 = pwm_lo_time;// for active low PWM driver
            interrupt_occurred = TRUE;

        }

    }
    else{
        if (PWM_LEFT_PIN == TRUE){
            PWM_LEFT_PIN = FALSE;
//            CMP01 = pwm_lo_time;// for active high PWM driver
            CMP01 = pwm_hi_time;// for active low PWM driver

        }
        else{
//            PWM_LEFT_PIN = TRUE;
            CMP01 = pwm_hi_time;// for active high PWM driver
            CMP01 = pwm_lo_time;// for active low PWM driver
            interrupt_occurred = TRUE;

        }

    }

}

// *****

void delay (unsigned char dly){

    while (--dly);

}

// *****
```

Facsimile Message

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

From:

Name

Company

Tel. FAX

Address

Thank you for your kind support.

North America NEC Electronics America Inc. Corporate Communications Dept. Fax: 1-800-729-9288 1-408-588-6130	Hong Kong, Philippines, Oceania NEC Electronics Hong Kong Ltd. Fax: +852-2886-9022/9044	Asian Nations except Philippines NEC Electronics Singapore Pte. Ltd. Fax: +65-6250-3583
Europe NEC Electronics (Europe) GmbH Market Communication Dept. Fax: +49(0)-211-6503-1344	Korea NEC Electronics Hong Kong Ltd. Seoul Branch Fax: 02-528-4411	Japan NEC Semiconductor Technical Hotline Fax: +81- 44-435-9608
	Taiwan NEC Electronics Taiwan Ltd. Fax: 02-2719-5951	

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Technical Accuracy	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

[MEMO]