RENESAS

Application Note

78K0

µPD78010x Subseries

Low Cost DC-Motor Control
using NEC 8-bit Single-Chip Microcontrollers

Document No. U17482EE1V0AN00
Date Published April 2005

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.

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

List of Figures

List of Tables

Chapter 1 Overview

1.1 Introduction

This application note describes a low-cost motor control application using an NEC µPD780103 microcontroller.

It covers the issues involved in driving and controlling a DC motor and is intended to help users to understand the dedicated motor control peripherals of the µPD780103 Subseries. The published software and hardware configurations are meant to serve as examples only and are not intended for mass production.

1.2 μPD780103 Overview

Table 1-1: Functional Outline

Item		Function
Internal memory	ROM	24 kbytes
	High-speed RAM	768 bytes
Memory space		64 kbytes
X1 input clock (oscillation frequency)		Ceramic/crystal/external clock oscillation 10 MHz (V_{DD} = 4.0 to 5.5 V) 8.38 MHz (V_{DD} = 3.3 to 5.5 V) 5 MHz (V_{DD} = 2.7 to 5.5 V)
Ring OSC clock (oscillation frequency)		On-chip ring oscillation (240 kHz (typ.))
General-purpose registers		8 bits × 32 registers (8 bits × 8 registers × 4 banks)
Minimum instruction execution time		0.2 µs/0.4 µs/0.8 µs/1.6 µs/3.2 µs (X1 input clock: @ f_{XP} = 10 MHz operation)
		8.3 µs/16.6 µs/33.2 µs/66.4 µs/132.8 µs (TYP.) (Ring OSC clock: @ f_R = 240 kHz (typ.) operation)
Instruction set		• 16-bit operation • Multiply/divide (8 bits × 8 bits, 16 bits ÷ 8 bits) • Bit manipulate (set, reset, test, and Boolean operation) • BCD adjust, etc.
I/O ports		Total 22 CMOS I/O 17 CMOS input 4 CMOS output 1
Timers		• 16-bit timer/event counter: 1 channel • 8 -bit timer/event counter: 1 channel • 8 -bit timer: 2 channels • Watchdog timer: 1 channel
	Timer outputs	4 (PWM: 3)
A/D converter		10-bit resolution × 4 channels
Serial interface		• UART mode supporting LIN bus: 1 channel • 3-wire serial I/O mode/UART mode*: 1 channel
Vectored interrupt sources	Internal	12
	External	6
Reset		• Reset using $\overline{\text{RESET}}$ pin • Internal reset by watchdog timer • Internal reset by clock monitor • Internal reset by power-on-clear • Internal reset by low-voltage detector
Supply voltage		V_{DD} = 2.7 to 5.5 V
Ambient operating temperature		T_A = -40 to +85 °C
Package		30-pin plastic SSOP (7.62 mm (300))

Note: * - Select either of these alternate-functions pins.

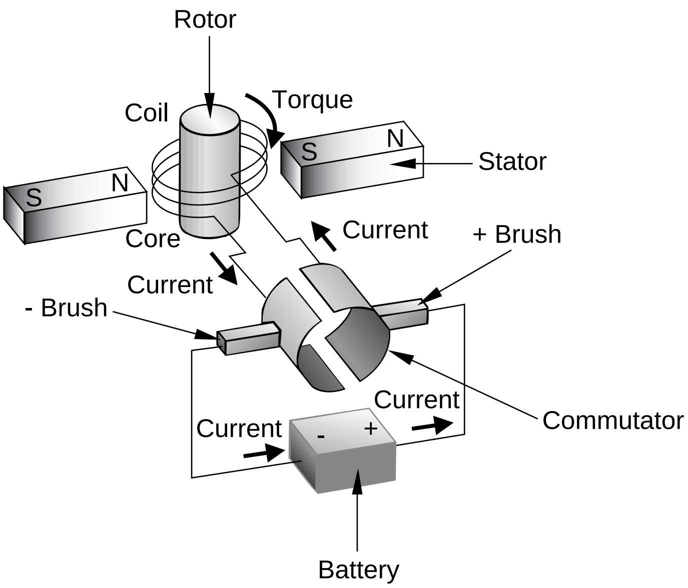
Chapter 2 DC-Motors

2.1 DC Motor Basics

DC motors are rotating electric machines designed to operate from source of direct voltage. They are inexpensive, easy to drive, and readily available in all shapes and sizes. As such, they are used widely in the industrial, automotive and consumer markets, e.g., for windshield wipers and fans.

A brushed DC motor is the simplest of all motor types, and typically consists a stator, rotor (armature), carbon brushes and commutator.

Figure 2-1: DC Motor - Operation



The stator of a brushed DC motor consists of two or more permanent magnets generating a stationary magnetic field that surrounds the rotor.

The rotor (armature) of a DC motor has coils of wire wrapped around its core. These coils are connected to small copper surface, called commutators, mounted on the rotor shaft. The rotor coils are energized by passing current through the carbon brushes that slide over the commutator segments.

The energized rotor coils produce a magnetic field. The opposite polarities of the rotor and stator field attract each other and the rotor starts to turn.

When the rotor and stator fields are aligned, the brushes move across the commutator segments and energize the next rotor winding.

The speed and torque of the motor depends on the strength of the magnetic field generated by the energized windings of the motor. The strength of the magnetic field depends on the strength of the current. Since speed is directly proportional to armature voltage and inversely proportional to the magnetic flux produced by the poles, adjusting the armature voltage and/or the field current changes the motor speed. In this application note speed control is based on generating and varying a PWM signal.

2.2 DC Motor Control Requirements

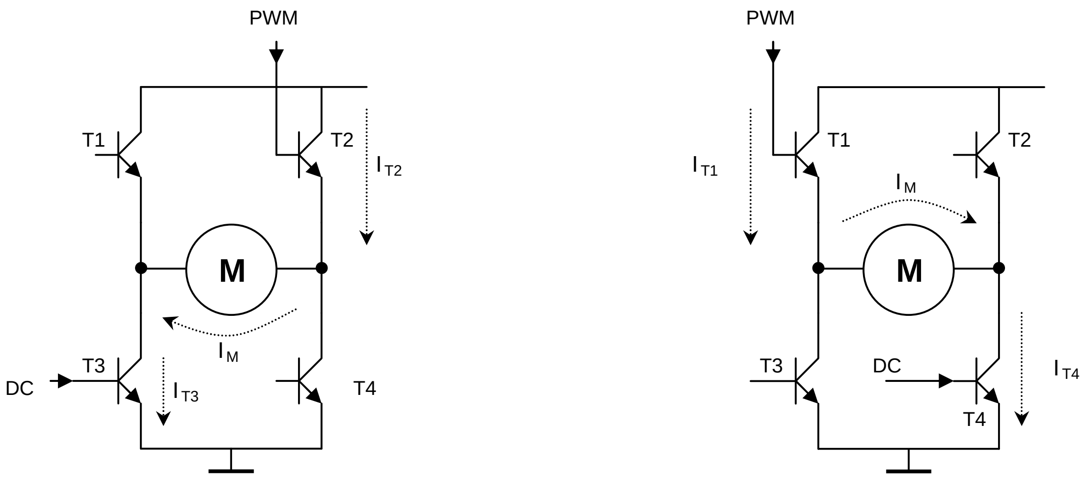
A popular circuit for driving DC motors is the H-bridge where four power switches, e.g., field effect transistors (MOSFET), are configured in an H pattern.
The H-bridge is used to change the sense of rotation, allowing a DC motor to run forward or backward with a single supply.

The motor speed is influenced by varying the duty cycle of the PWM signal. Varying the duty cycle changes the average DC voltage that is used to drive the power MOSFETs.

Figure 2-2 shows the basic principle.

Figure 2-2: H-bridge

H-bridge (forward) **H-bridge (backward)**



The H-bridge consists of two of high-side (T1, T2) and of two low-side (T3, T4) transistors. To drive the motor in the forward direction, one high-side transistor (T2) is controlled with PWM, and one low-side transistor (T3) is controlled by DC (soft chopping). T1 and T4 are turned off.

If the PWM pulse is high, T2 is open, and the current flows through T2, over the motor and through T3 to ground (Fig. 2-2a). If the PWM pulse is low, T2 is turned off, and the motor is slowed down. Changing the duty cycle changes the motor speed.

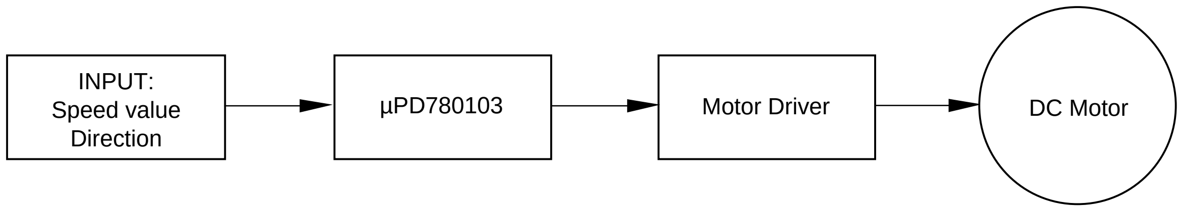
To drive the motor in the reverse direction, T1 is controlled with PWM, and T4 is controlled by DC. T2 and T3 are turned off. The current flows in reverse direction and the motor turns in the opposite direction (Fig. 2-2b).

Chapter 3 System Concept

3.1 System Design

Figure 3-1 shows the basic block diagram for the DC motor control.

Figure 3-1: Basic System Configuration

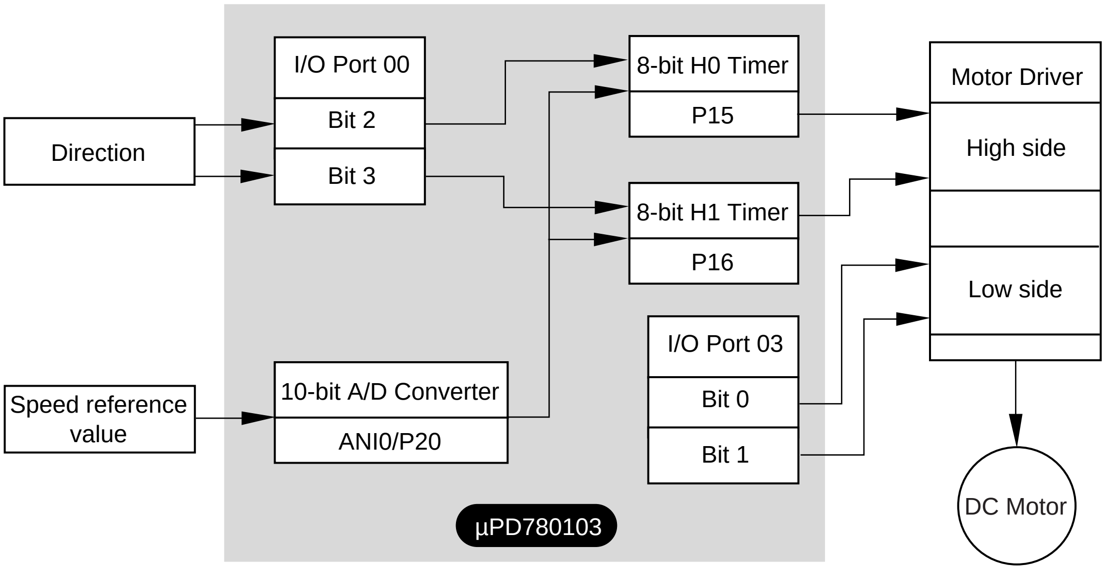


The µPD780103 evaluates the input signal for sense of rotation and motor speed. The µPD780103 uses this information to generate the appropriate control signals for the motor driver.

3.2 System Configuration

Figure 3-2 shows the system configuration and the µPD780103 peripherals used for the DC motor control.

Figure 3-2: System Configuration with µPD780103 Peripherals



Bit 2 and bit 3 of port 00 are used to scan the switch position that defines the sense of rotation of the motor. The H0 or H1 timers are activated to generate a PWM signal.
The set voltage on the potentiometer is proportional to the desired motor speed. This voltage is converted by the 10 bit A/D converter and then normalized to 8 bits. The normalized A/D value defines the duty cycle of the PWM signal that is generated by the H0 or H1 timer. Bit 0 and bit 1 of port 03 are used to control the low-side driver.

The function of each of the peripherals is described in the next chapter.

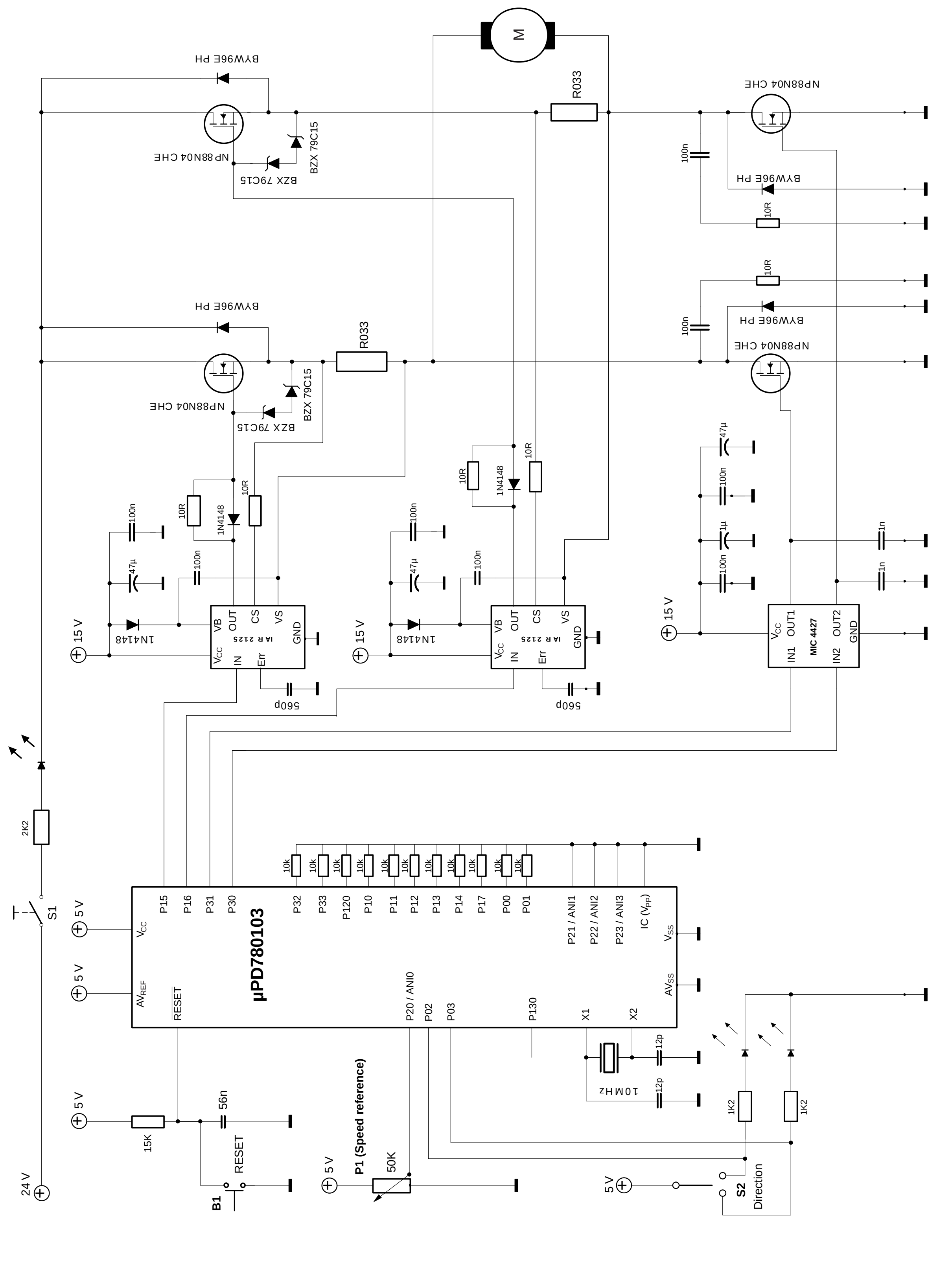
Chapter 4 Hardware Configuration

4.1 Schematic

The DC motor in this application has a rated speed of 3000 rpm @ 24 V.

The diagram shows the DC motor control design using a µPD780103 microcontroller and NEC's NP88N004 CHE PowerMOSFET.

Figure 4-1: Schematic Diagram



4.2 µPD780103 Configuration

The µPD780103 device is a member of the high-performance 78K 8-bit microcontroller family, designed specifically for mid-range motor control applications. The configuration of the device and the operating environment used in this application are listed below:

- CPU µPD780103
- Operating clock System clock – 10 MHz
- Operating voltage 5 V
- Internal ROM 24 Kbytes
- Internal RAM 768 bytes

4.3 Peripherals and I/O Usage

Table 4-2 shows the microcontroller's I/O pins.
The pins used in the application and their functions are listed.

Table 4-1: I/O Configuration

Pin number	Pin name	Signal name	Mode setting	Function
1	P33/INTP4	I/O	Input	Not used
2	P32/INTP3	I/O	Input	Not used
3	P31/INTP3	I/O	Output	Low-side control
4	P30/INTP1	I/O	Output	Low-side control
5	IC	V_{PP}		Not used
6	V_{SS}		Ground	Ground
7	V_{DD}		V_{DD}	V_{DD}
8	X1		Input	Main crystal
9	X2			Main crystal
10	$\overline{\text{RESET}}$		Input	System reset
11	P03	I/O	Input	Different direction
12	P02	I/O	Input	Different direction
13	P01/TI010/TO00	I/O	Input	Not used
14	P00/TI000	I/O	Input	Not used
15	P10/SCK10/TXD0	I/O	Input	Not used
16	P11/SI10/RXD0	I/O	Input	Not used
17	P12/SO12	I/O	Input	Not used
18	P13/TXD6	I/O	Input	Not used
19	P14/RXD6	I/O	Input	Not used
20	P15/TOH0	TOH0	Output	PWM out for high side
21	P16/TOH1/INTP5	TOH1	Output	PWM out for high side
22	P17/TI50/TO50	I/O	Input	Not used
23	P130	Output	Output	Not used
24	P23/ANI3	Input	Input	Not used
25	P22/ANI2	Input	Input	Not used
26	P21/ANI1	Input	Input	Not used
27	P20/ANI0	ANI0	Input	Speed reference
28	AV_{REF}	5 V		A/D reference
29	AV_{SS}	0 V		A/D ground
30	P120/INTP0		Input	Not used

4.4 8-bit Hn Timer

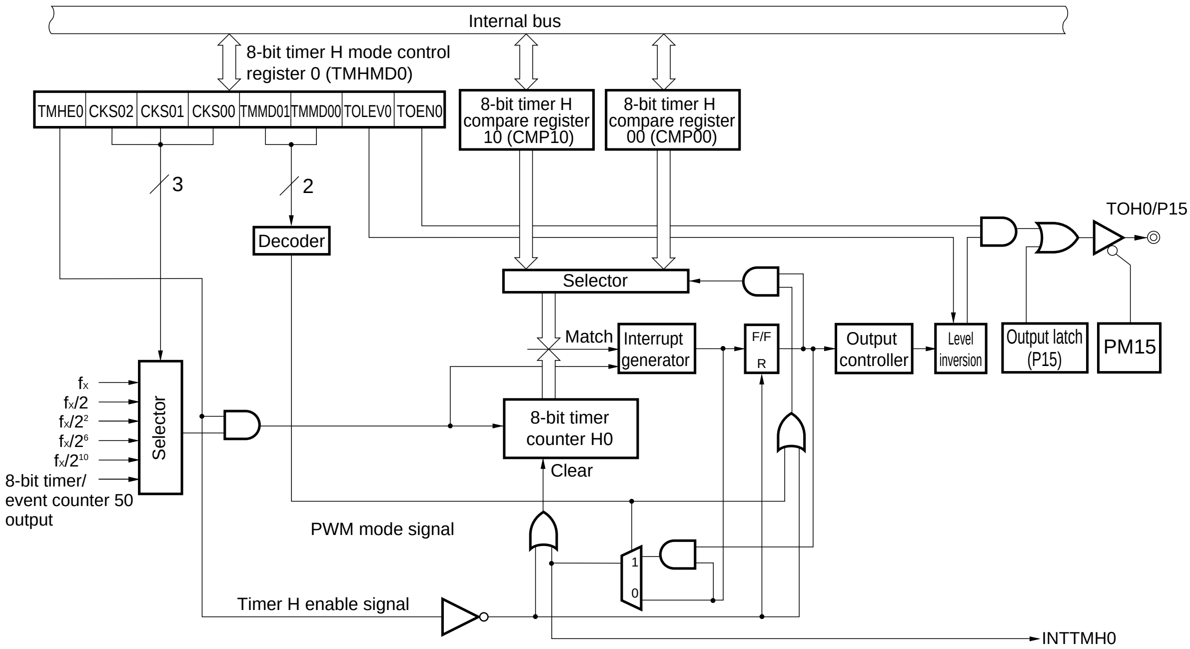
This timer is used to generate a PWM signal that controls the speed of the motor via the MOSFETS driver and MOSFET. The PWM signal is varied by modifying the duty cycle.

Hn (n = 0, 1) has the following functions:

- Interval timer
- Square-wave output
- PWM output

The following shows the block diagram of the H0 timer.

Figure 4-2: Block Diagram of 8-bit H0 Timer

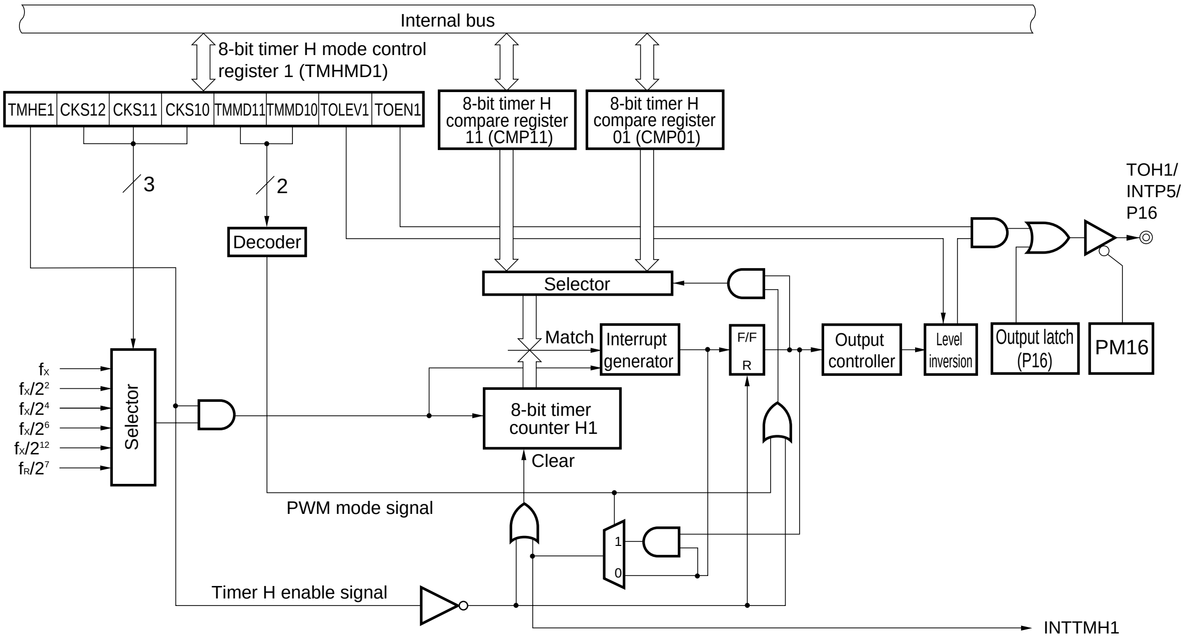


PWM output mode with a frequency of 9.6 kHz is selected. The duty ratio of the 9.6 kHz signal is determined by the value of the 8-bit timer CMP1n compare register.

The calculation result of the speed control algorithm can be set on the fly, i.e., the CMP1n register can be set at any time without stopping the Hn timer, because rewriting the CMP1n register is permitted during timer operation.

The Hn timer compare register CMP1n is used to modify the duty cycle of PWM.

Figure 4-3: Block Diagram of 8-bit H1 Timer

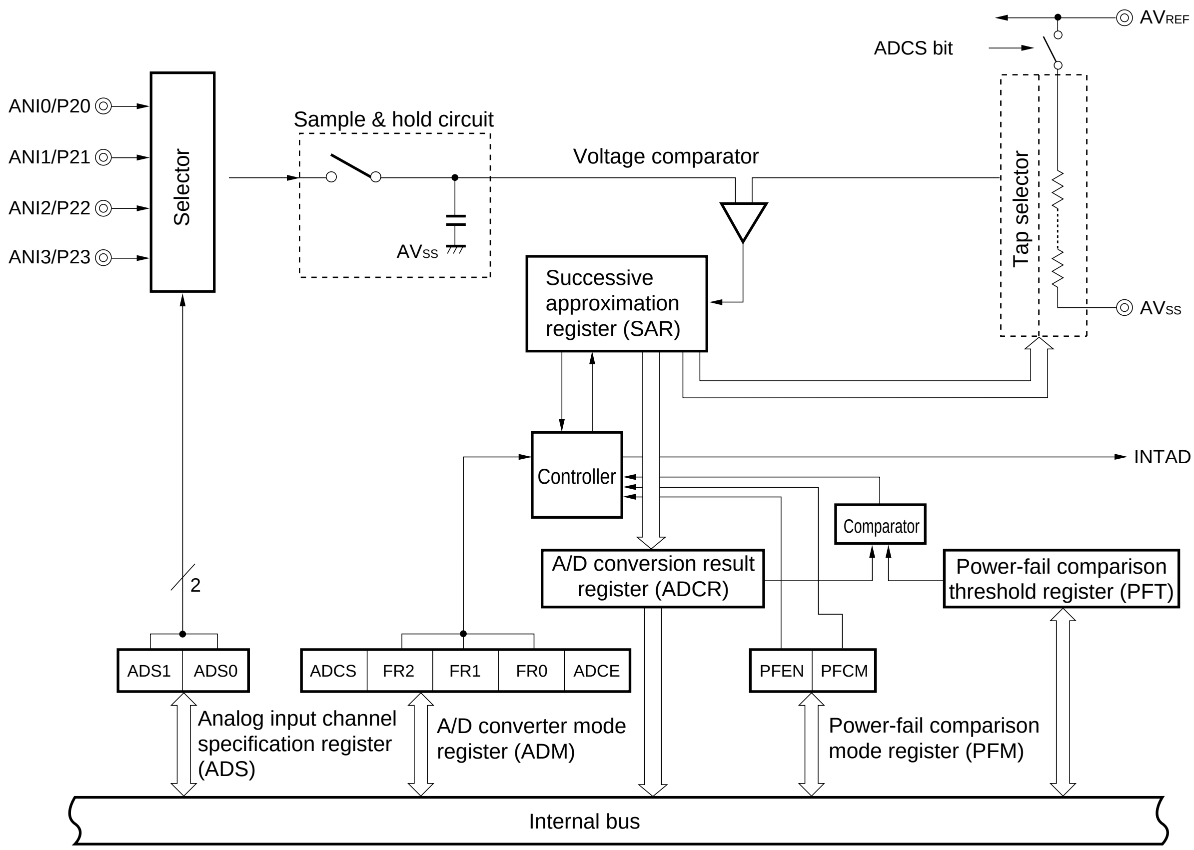


The H0 and H1 timers are configured identically.

4.5 A/D Converter

The µPD780103 has on-chip A/D converter with four input channels that converts an analog input signal into a digital value with 10-bit resolution.

Figure 4-4: Block Diagram of A/D Converter



In this application a potentiometer is connected between V_{DD} (5 V), ground and channel 1 (ANI0) of the A/D converter. The potentiometer value represents the reference value for the speed control.

The conversion process is continuous and always the last converted value is read out from the ADCR register of the µPD780103.

The conversion time of the A/D converter is 9.6 µs.

Analog reference AV_{DD} and analog ground of the A/D converter are connected to V_{DD} and V_{SS}, respectively.

[MEMO]

Chapter 5 Software Process Descriptions

5.1 Introduction

The software consists of four main sections:

<1> Main
<2> Control with 8-bit H0 timer
<3> Control with 8-bit H1 timer
<4> A/D ISR

The motor operation is controlled by the H0, H1 timers and the A/D converter interrupt service routine.

The main program consists of an initialization routine and start-up of the control process.

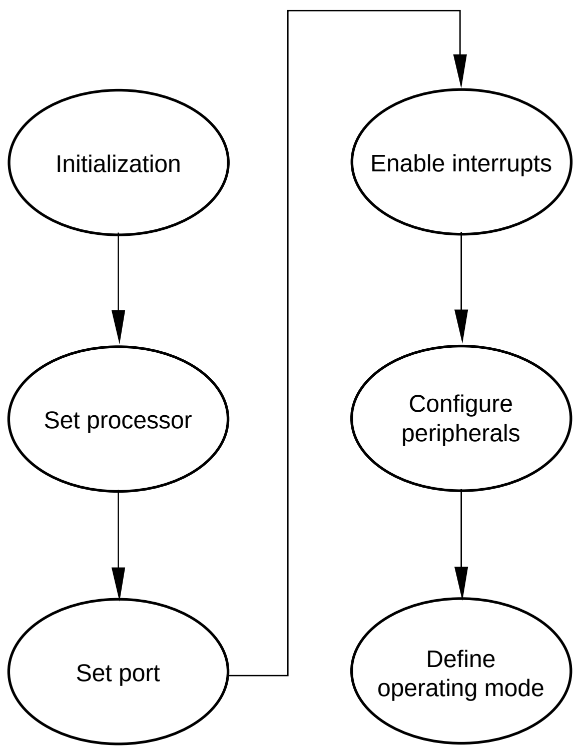
H0 and H1 generate the PWM signal for controlling the motor speed. The A/D converter ISR provides the speed reference value.

The operation of each of the software components is described in the following sections.

5.2 Initialization

The initialization routine is responsible for initializing the µPD780103 device after a system reset. It configures the clock settings of the device, initializes the peripherals used for the motor control application and enables the appropriate interrupts. The initialization has two parts as shown in Figure 5-1. The first part initializes and sets the processor clock of the device and the second part initializes the peripherals and their operating modes.

Figure 5-1: Initialization Process



The initialization contains the following sections:

<1> System clock setting
<2> 8-bit H0 timer
<3> 8-bit H1 timer
<4> A/D converter
<5> I/O port setting

5.3 Main

The main program provides the framework within which the subroutines execute the various tasks of the motor control application.

The main program starts by invoking the global initialization subroutines CPU_ClockTo_X1 and hdw_init.

The endless loop of the main program consists of the PWM generation, direction control and speed stabilization.

5.4 AD_convert (ISR)

The reference speed is given by a potentiometer.
The A/D converter continuously converts its input value. Approx. every 9.6 µs a new reference speed value is provided to the ADCR register for further calculations.
The ISR normalizes the A/D converter's 10-bit result to 8 bits. The ISR also calculates the difference between the old and new value of ADCR register. Both values are passed to appropriate routines for further handling.

5.5 Ramp

This routine performs a soft start-up of the DC motor and soft variance of the motor speed.

After determining of the speed value, the PWM register of the Hn timer is set to minimum speed. Hn generates the PWM signal for controlling the motor speed.

PWM generation of Hn is enabled and the PWM register, containing the minimum speed value, is incremented to increase the motor speed until the reference speed is reached.

Remark: n = 0, 1

5.5.1 CMP_reg

This routine is invoked from within the ramp routine.

For a soft start-up or soft speed change, each change in the PWM duty cycle is followed by a delay.
This routine also determines which timer register must be set to the actual value.

5.6 turn_left or turn_right

This is the main control program and consists of the two routines turn_left() and turn_right() that provide the following functions:

(a) Set the PWM frequency
(b) Initialize the port that controls the low-side driver
(c) Invoke the ramp function
(d) Determine the direction

Chapter 6 Conclusion

This application example provides a basic DC motor control system suitable for implementation using NEC's 78K0 8-bit microcontroller family.

Memory requirements for this example:

- 78K0 – µPD780103

Code: 24 kbytes available, 1869 bytes used
Data: 768 bytes available, 136 bytes used

The 78K0 microcontroller used in this application provides further resources for additional user functions or/and adaptations of the control algorithm.

Chapter 7 Software Flow Charts

Figure 7-1: Main Program Flowchart

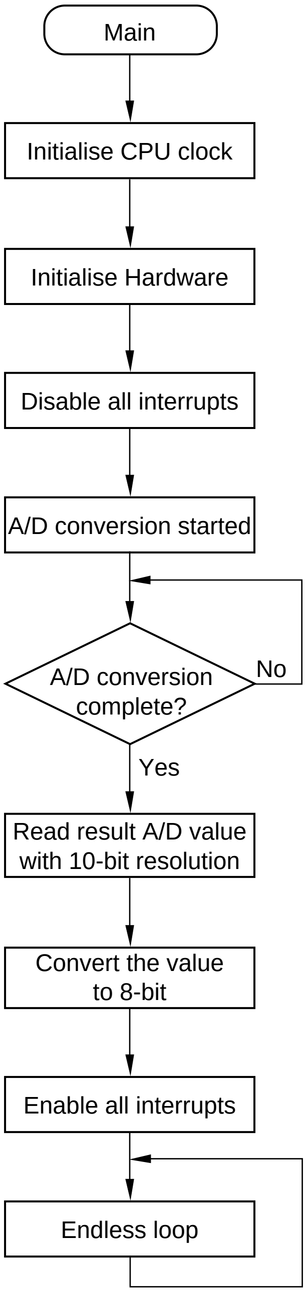


Figure 7-2: CPU Clock Initialization

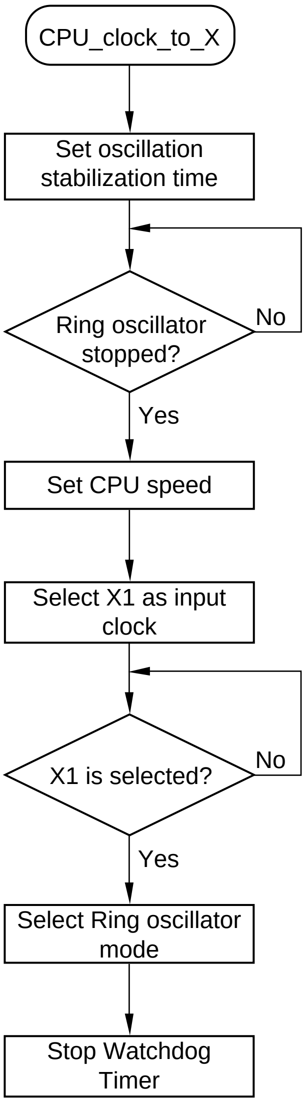


Figure 7-3: Hardware Initialization

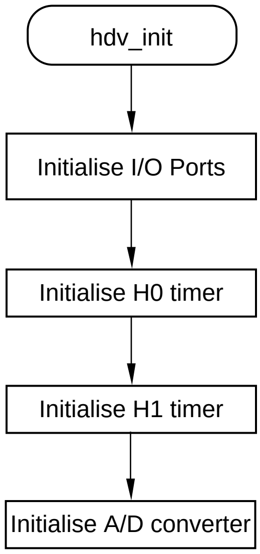


Figure 7-4: Main Endless Loop

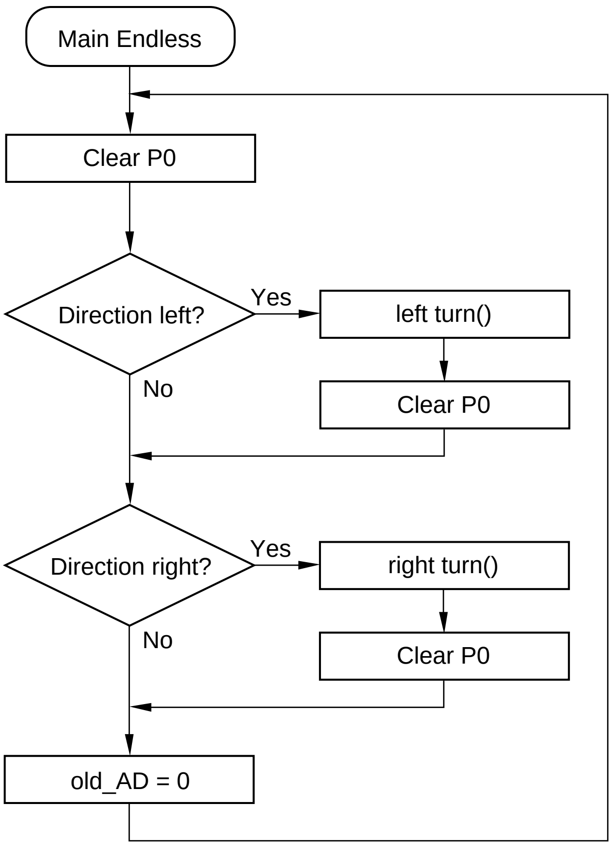


Figure 7-5: AD_convert (ISR)

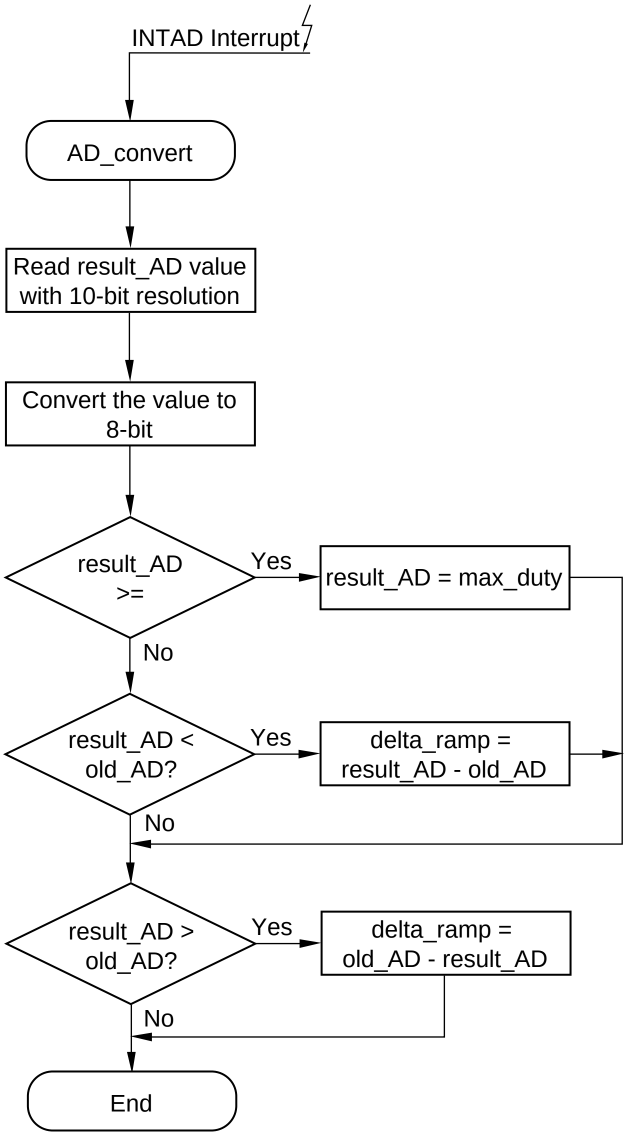


Figure 7-6: Left_turn()

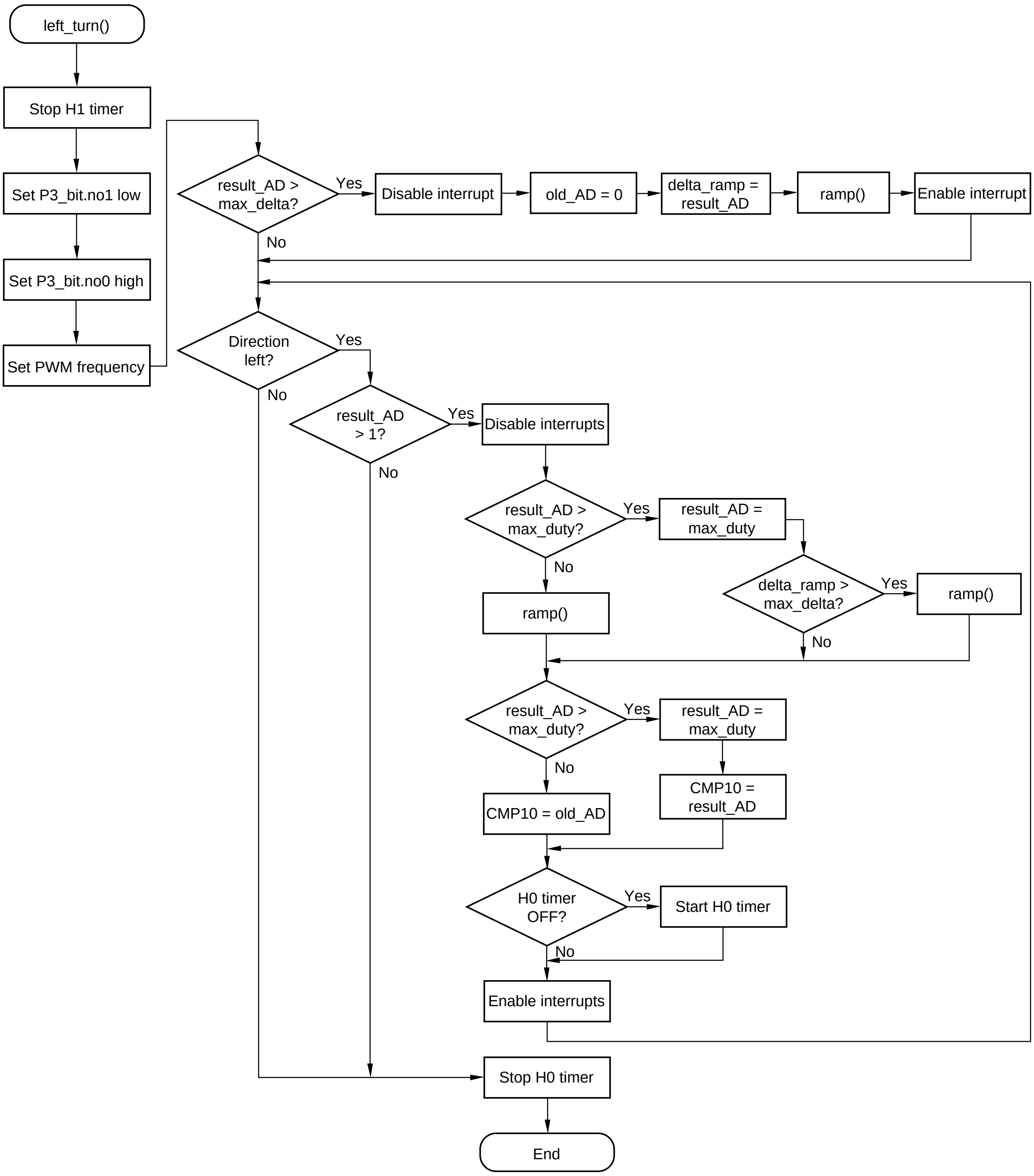


Figure 7-7: Right_turn()

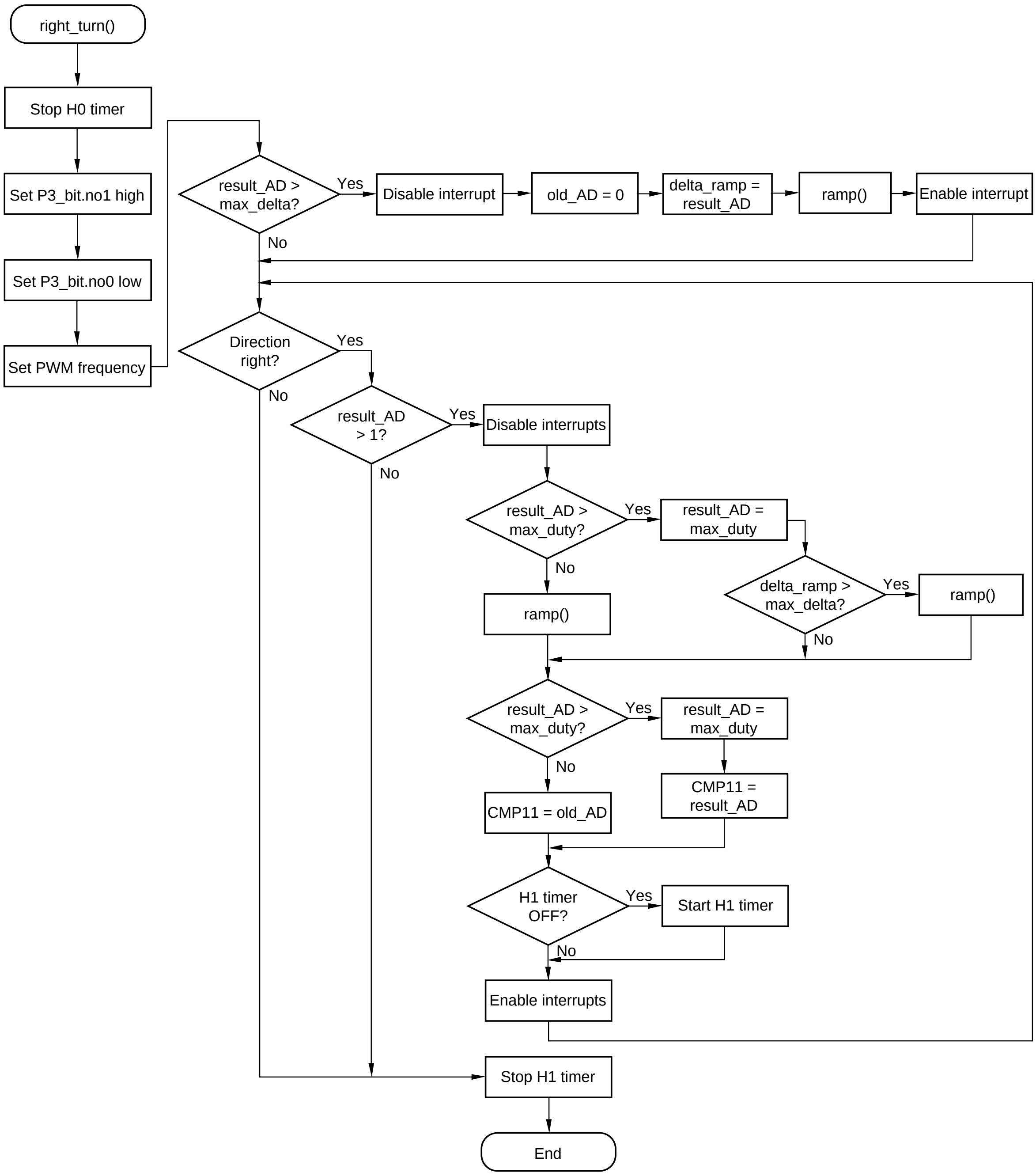


Figure 7-8: Ramp()

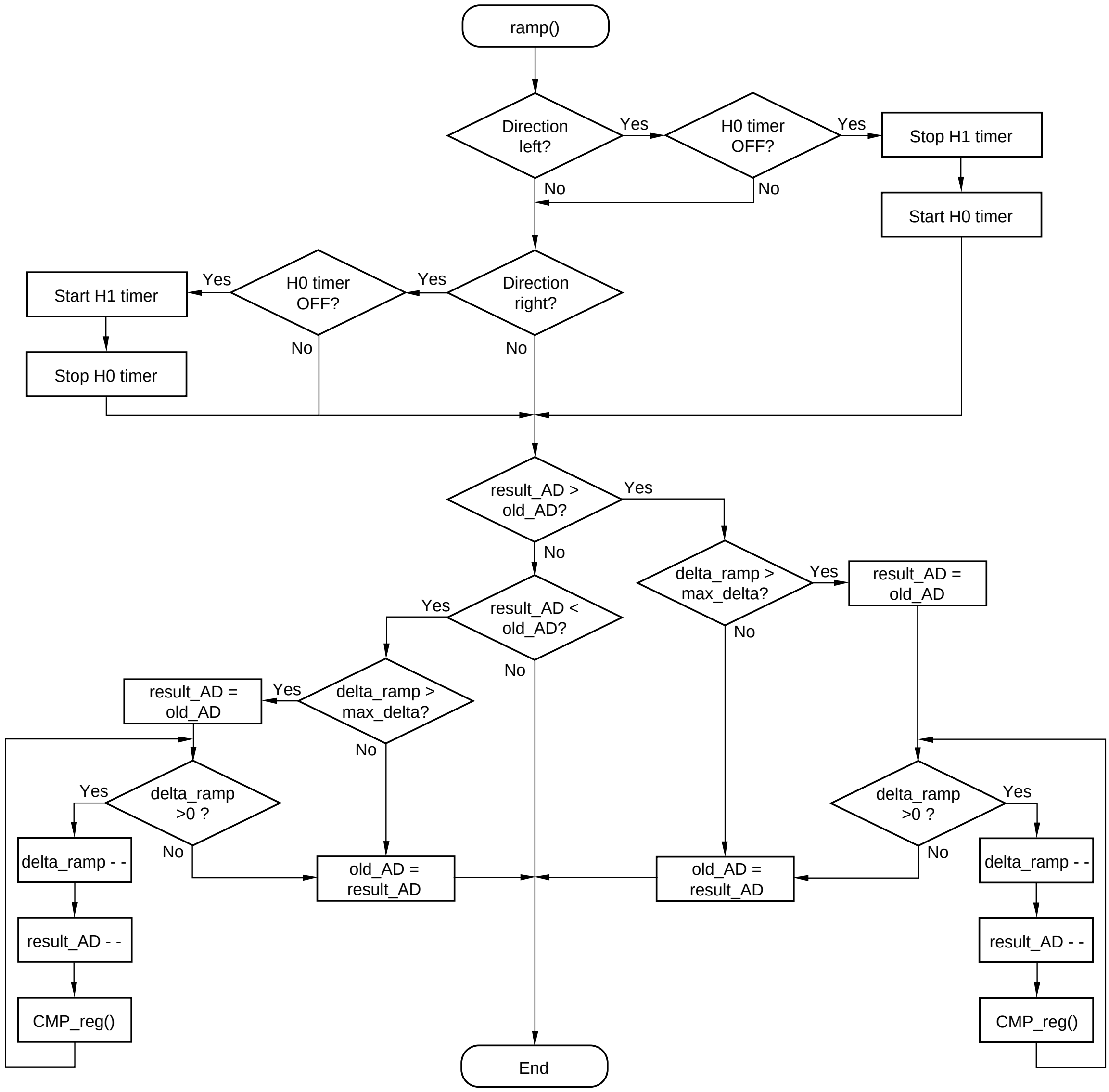


Figure 7-9: CMP_reg()

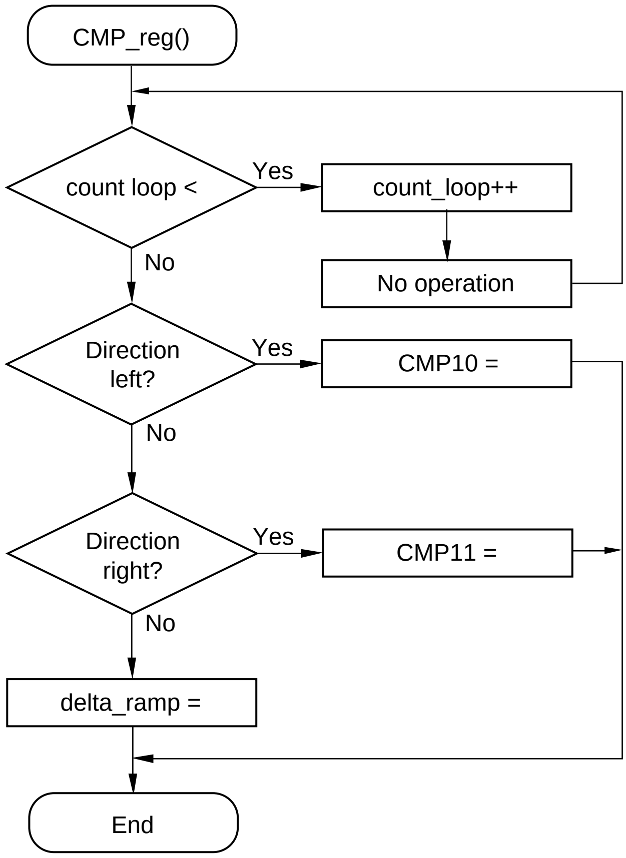


Figure 7-10: Disable all Interrupts

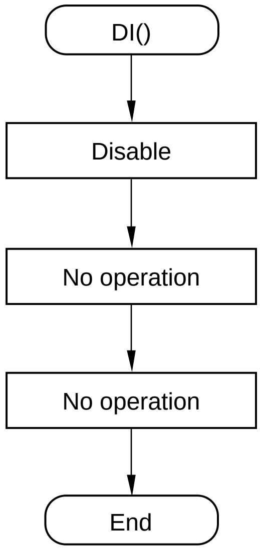
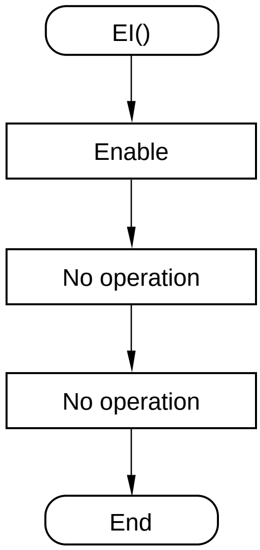


Figure 7-11: Enable all Interrupts



[MEMO]

Chapter 8 Program Listing

```
/*********************************************************************/
/* PROJECT    = Motor control                                        */
/* MODULE     = declar.c                                             */
/* VERSION    = V0.1                                                 */
/* DATE       = 18.02.2005                                           */
/* LAST CHANGE=                                                      */
/*                                                                   */
/* NEC Electronics (Europe) GmbH                                     */
/* Technical Product Support                                         */
/* Customer Engineering Support                                      */
/* D-40472 Düsseldorf, Germany                                       */
/*********************************************************************/
/* Description: global declaration                                   */
/*********************************************************************/

volatile unsigned short result_AD, delta_ramp;
unsigned short old_AD;
unsigned short count_loop;

/*********************************************************************/


/*********************************************************************/
/* PROJECT    = Motor control                                        */
/* MODULE     = declar.h                                             */
/* VERSION    = V0.1                                                 */
/* DATE       = 18.02.2005                                           */
/* LAST CHANGE=                                                      */
/*                                                                   */
/* NEC Electronics (Europe) GmbH                                     */
/* Technical Product Support                                         */
/* Customer Engineering Support                                      */
/* D-40472 Düsseldorf, Germany                                       */

/*********************************************************************/
/* Description: global declaration and definition                    */

/*********************************************************************/


#ifndef __DECLAR_H__
#define __DECLAR_H__

#define max_duty 235    // define maximum of duty cycle
#define PWM_FQ   255    // define PWM frequency
#define max_delta 10    // define maximum of delta
#define delay_val 3500  // define maximum of delay time value

extern volatile unsigned short result_AD, delta_ramp;
extern unsigned short old_AD;
extern unsigned short count_loop;
#endif                  //__DECLAR_H__

/*********************************************************************/
```

```
/***********************************************************************/
/* PROJECT    = Motor control                                          */
/* MODULE     = init.c                                                 */
/* VERSION    = V0.1                                                   */
/* DATE       = 18.02.2005                                             */
/* LAST CHANGE=                                                        */
/*                                                                     */
/* NEC Electronics (Europe) GmbH                                       */
/* Technical Product Support                                           */
/* Customer Engineering Support                                        */
/* D-40472 Düsseldorf, Germany                                         */
/***********************************************************************/
/* Description: Module initializes the UPD780103 Hardware              */
/***********************************************************************/


/***********************************************************************/
/* include                                                             */
/***********************************************************************/
#include 'io780103.h'
#include <intrinsics.h>

/***********************************************************************/
/* pragma                                                              */
/***********************************************************************/
#pragma language = extended
/***********************************************************************/
/* type definitions (function prototypes)                              */
/***********************************************************************/
void CPU_ClockTo_X1(void);          // Select CPU clock
void hdw_init (void);               // Hardware initialization
/***********************************************************************/

void CPU_ClockTo_X1(void)
{
    OSTS = 0x02;                    // 6.55ms@10MHz wait after STOP
                                    // mode release (RESET value)
                                    // -----|||   Osc. stabilization time
                                    // -----010 - 2^13/fx

while (!OSTC_bit.no3) {__no_operation();}
                                    // wait until ring oscillator stopped
 PCC = 0x00;                        // Use high speed mode fCPU=10MHz
 MCM = 0x01;                        // ------|1 - X1 input clock
 MOC = 0x00;                        // 0 -------- X1 input clock oscillating

while (!MCM_bit.no1)
{
 MOC = 0x00;
 MCM = 0x01;
}

 RCM = 0x01;                        // Ring oscillator stopped
 WDTM = 0x77;                       // 01110111 watchdog timer stopped

}                                   // end of vSwitchCPUClockToX1()
```

```
/*********************************************************************/

void hdw_init (void)
{

/*********************************************************************/
/*************   setting port mode register, port register   ************/

PM0=0xFF;                  //1111 1111 P02,P03,P00,P01 as input
PM1=0x9F;                  //1001 1111 P15, P16 as output
PM3=0xFC;                  //1111 1100 P30, P31 as output for low side

P0=0x00;                   //0000 0000 P02,P03,P00,P01 to input low level
P1=0x00;                   //0000 0000 P15,P16 to output latch
P3=0x00;                   //0000 0000 P30-P33 to output latch

/*********************************************************************/
__enable_interrupt();
/*********************************************************************/

/************************     Timer H0, H1 setting    *****************/

TMHMD0=0x29;               //0010 1001 f=2,5M;PWM mode;stopped;output enabled
TMHMD1=0x19;               //0001 1001 f=2,5M;PWM mode;stopped;output enabled

/*********************************************************************/
/*****************     A/D Converter setting    ***********************/
ADS=0x00;                  //00000000 ANI0 as input
ADM=0x31;                  //00110001 operation with 9.6µs and with
                           //AVref (stopped)
ADMK=0;                    //interrupt for AD-conversion enabled
/*********************************************************************/
}                          //end of hdw_init()
```

```
/**********************************************************************/
/* PROJECT     = Motor control                                        */
/* MODULE      = main.c                                               */
/* VERSION     = V0.1                                                 */
/* DATE        = 18.02.2005                                           */
/* LAST CHANGE=                                                       */
/*                                                                    */
/* NEC Electronics (Europe) GmbH                                      */
/* Technical Product Support                                          */
/* Customer Engineering Support                                       */
/* D-40472 Düsseldorf, Germany                                        */
/**********************************************************************/
/* Description: Main function                                         */
/**********************************************************************/


/**********************************************************************/
/* type definitions (function prototypes)                             */
/**********************************************************************/
void CPU_ClockTo_X1(void);       // Select CPU clock
void hdw_init (void);            // Hardware initialization
void left_turn(void);            // Motorcontrol with H0 timer
void right_turn(void);           // Motorcontrol with H1 timer
void ramp(void);                 // ramp function
void DI(void);                   // disable interrupts
void EI(void);                   // enable interrupts
/**********************************************************************/
// pragma
/**********************************************************************/
#pragma language = extended

/**********************************************************************/
// include
/**********************************************************************/
#include 'io780103.h'
#include 'declar.h'
#include <intrinsics.h>
/**********************************************************************/


void main(void)
{
 CPU_ClockTo_X1();
 hdw_init ();

 DI();                           // disable interrupts

 ADCS=1;                         // start conversion
 while(!ADIF){}                  // wait for conversion
 result_AD=(ADCR>>8)-0.5;        // Convert the 10 bit Result register
                                 // to 8 bit and to integer

 EI();                           // enable interrupts
```

```
 while (1)                        // start endless loop
  {
   P0=0x00;                       // Clear P0

    if (P0_bit.no2 == 1)          // turn left
    {
      left_turn();
      P0=0x00;                    // Clear P0
    }                             // end of     if (P0_bit.no2 == 1)



    if (P0_bit.no3 == 1)          // turn right
    {
      right_turn();
      P0=0x00;                    // Clear P0
    }                             // end of     if (P0_bit.no3 == 1)

    old_AD=0;                     // new initialisation of old value of ADC

 }                                // end of while(1)
}                                 // end of main
```

```
/***********************************************************************/
/* PROJECT     = Motor control                                        */
/* MODULE      = interr.c                                             */
/* VERSION     = V0.1                                                 */
/* DATE        = 18.02.2005                                           */
/* LAST CHANGE=                                                       */
/*                                                                    */
/* NEC Electronics (Europe) GmbH                                      */
/* Technical Product Support                                          */
/* Customer Engineering Support                                       */
/* D-40472 Düsseldorf, Germany                                        */
/***********************************************************************/
/* Description: interrupt vector                                      */
/***********************************************************************/
/***********************************************************************/
// include
/***********************************************************************/
#include 'io780103.h'
#include <intrinsics.h>
#include <declar.h>
/***********************************************************************/

#pragma vector = INTAD_vect

__interrupt void AD_convert(void)
{
  result_AD=(ADCR>>8)-0.5;

  if(result_AD>=max_duty)
  {
     result_AD=max_duty;         //set limit of duty cycle
  }

  if(result_AD < old_AD)
   {
   delta_ramp = old_AD - result_AD;
   }
  else if(result_AD > old_AD)
   {
   delta_ramp = result_AD - old_AD;
   }

}                                //end of AD_convert()
```

```
/*********************************************************************/
/* PROJECT     = Motor control                                       */
/* MODULE      = ramp.c                                              */
/* VERSION     = V0.1                                                */
/* DATE        = 18.02.2005                                          */
/* LAST CHANGE=                                                      */
/*                                                                   */
/* NEC Electronics (Europe) GmbH                                     */
/* Technical Product Support                                         */
/* Customer Engineering Support                                      */
/* D-40472 Düsseldorf, Germany*/
/*********************************************************************/
/* Description: ramp function                                        */
/*********************************************************************/

/*********************************************************************/
/* type definitions (function prototypes)                            */
/*********************************************************************/
void ramp(void);                 // ramp function
void CMP_reg(void);              // determine the CMP-register
/*********************************************************************/
// pragma
/*********************************************************************/
#pragma language = extended

/*********************************************************************/
// include
/*********************************************************************/
#include 'io780103.h'
#include 'declar.h'
#include <intrinsics.h>
/*********************************************************************/
void ramp(void)
{

if((P0_bit.no2 == 1)&&(!TMHE0))
{
 TMHE1=0;                        //stop H1
 TMHE0=1;                        //start H0
}
else if ((P0_bit.no3 == 1)&&(!TMHE1))
{
 TMHE0=0;                        //stop H0
 TMHE1=1;                        //start H1
}

  if(result_AD > old_AD)
   {

      if(delta_ramp > max_delta)
       {
        result_AD = old_AD;
        for (delta_ramp; delta_ramp > 0; delta_ramp--)
          {
           result_AD++;
           CMP_reg();
          }                      //end of for delta_ramp
```

```
        }                        //end if(delta_ramp > max_delta)

        old_AD = result_AD;
     }                           //end of if(result_AD > old_AD)
/************************************************************************/
  else if(result_AD < old_AD)
   {

    if(delta_ramp > max_delta)
     {
      result_AD = old_AD;
      for (delta_ramp; delta_ramp > 0; delta_ramp--)
       {
         result_AD--;
         CMP_reg();
        }                        //end of for delta_ramp
      }                          //end of if(delta_ramp > max_delta)

      old_AD = result_AD;
    }                            //end of if(result_AD < old_AD)
/************************************************************************/

}                                //end of function ramp()


/************************************************************************/
/************************************************************************/

void CMP_reg(void)
{
    for(count_loop=0;count_loop<delay_val;count_loop++)//time delay
      {
        __no_operation();
      }



    if(P0_bit.no2 == 1)
      {
       CMP10 = result_AD;
      }
     else if (P0_bit.no3 == 1)
      {
       CMP11=result_AD;
      }
     else
      {
       delta_ramp=1;//break of for-loop
      }
}                                // end of function CMP_reg()
```

```
/***********************************************************************/
/* PROJECT     = Motor control                                         */
/* MODULE      = control.c                                             */
/* VERSION     = V0.1                                                  */
/* DATE        = 18.02.2005                                            */
/* LAST CHANGE=                                                        */
/*                                                                     */
/* NEC Electronics (Europe) GmbH                                       */
/* Technical Product Support                                           */
/* Customer Engineering Support                                        */
/* D-40472 Düsseldorf, Germany                                         */
/***********************************************************************/
/* Description: Motor control with H0 and H1 timers                    */
/***********************************************************************/


/***********************************************************************/
// include
/***********************************************************************/
#include 'io780103.h'
#include 'declar.h'
#include <intrinsics.h>
/***********************************************************************/


/***********************************************************************/
/* type definitions (function prototypes)                              */
/***********************************************************************/
void left_turn(void);           // Motorcontrol with H0 timer
void right_turn(void);          // Motorcontrol with H1 timer
void ramp(void);                // ramp function
void DI(void);                  // disable interrupts
void EI(void);                  // enable interrupts
/***********************************************************************/



/*******************     Motorcontrol with H0 timer    *****************/
void left_turn (void)
{

     TMHE1=0;                   // stop H1
     P3_bit.no1 = 0;            // Low Side of H1 OFF
     P3_bit.no0 = 1;            // Low Side of H0 ON
     CMP00=PWM_FQ;              // Set PWM frequency for H0 = 9.6 kHz

      if(result_AD>max_delta)
      {
       DI();                    // disable interrupts

        old_AD=0;
        delta_ramp = result_AD;
        ramp();

        EI();                   // enable interrupts
      }
```

```
      while ((P0_bit.no2 == 1)&&(result_AD >=1))
       {

       DI();                      // disable interrupts

        if(result_AD>=max_duty)
        {
          result_AD = max_duty;  // set limit of duty cycle
          if(delta_ramp > max_delta)
           {
            ramp();
           }
        }
        else
        {
          ramp();
        } // end of else


        if(result_AD>=max_duty)
        {
          result_AD = max_duty;  // set limit of duty cycle
          CMP10 = result_AD;
        }
        else
        {
        CMP10 = old_AD;
        }


       if (!TMHE0)
       {
         TMHE0=1;                 // start H0
       }

       EI();                      // enable interrupts

       }//End of while


      TMHE0=0;                    // stop H0


}// end of left_turn()
```

```
/*********************************************************************/
/*****************     Motorcontrol with H1 timer    *******************/
void right_turn(void)
{

     TMHE0=0;                    // stop H0
     P3_bit.no1 = 1;             // Low Side of H1 ON
     P3_bit.no0 = 0;             // Low Side of H0 OFF
     CMP01=PWM_FQ;               // Set PWM frequency for H0 = 9.6 kHz

      if(result_AD>max_delta)
      {
       DI();                     // disable interrupts

        old_AD=0;
        delta_ramp = result_AD;
        ramp();

        EI();                    // enable interrupts
      }


      while ((P0_bit.no3 == 1)&&(result_AD >=1))
       {

       DI();                     // disable interrupts

        if(result_AD>=max_duty)
        {

          result_AD=max_duty;    // set limit of duty cycle
          if(delta_ramp > max_delta)
           {
            ramp();
           }
        }
        else
        {
          ramp();
        }                        // end of else


        if(result_AD>=max_duty)
        {
          result_AD = max_duty;  // set limit of duty cycle
          CMP11 = result_AD;
        }
        else
        {
        CMP11 = old_AD;
        }


       if (!TMHE1)
       {
         TMHE1=1;                // start H1
       }
```

```
       EI();                      // enable interrupts
       }//end of while
      TMHE1=0;                        //stop H1
}                                 // end of right_turn()
/************************************************************************/
```

```
/************************************************************************/
/* PROJECT    = Motor control                                          */
/* MODULE     = d_e_inter.c                                            */
/* VERSION    = V0.1                                                   */
/* DATE       = 18.02.2005                                             */
/* LAST CHANGE=                                                        */
/*                                                                     */
/* NEC Electronics (Europe) GmbH                                       */
/* Technical Product Support                                           */
/* Customer Engineering Support                                        */
/* D-40472 Düsseldorf, Germany                                         */
/************************************************************************/
/* Description: Disable or enable interrupts                           */
/************************************************************************/

/************************************************************************/
/* type definitions (function prototypes)                              */
/************************************************************************/
void DI(void);
void EI(void);
/************************************************************************/

/************************************************************************/
// include
/************************************************************************/
#include <intrinsics.h>
/************************************************************************/

void DI(void)

{

__disable_interrupt();
__no_operation();
__no_operation();

}

/************************************************************************/

void EI(void)

{

__enable_interrupt();
__no_operation();
__no_operation();

}

/************************************************************************/
```

[MEMO]

NEC

Facsimile Message

From:

Name

Company

Tel. FAX

Address

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.

Thank you for your kind support.

North America
NEC Electronics America Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Market Communication Dept.
Fax: +49(0)-211-6503-1344

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-2719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-6250-3583

Japan
NEC Semiconductor Technical Hotline
Fax: +81- 44-435-9608

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	☐	☐	☐	☐
Technical Accuracy	☐	☐	☐	☐
Organization	☐	☐	☐	☐

CS 99.1

[MEMO]