King Mongkut's University of Technology Thonburi
Department of Electronics and Telecommunication Engineering Faculty of Engineering
EIE/ENE 335 Digital Circuit and Microprocessor Lab for the 3rd year student

Experiment: DC-motor

Objectives

- How to use
  - the NuMicro™ NUC100 series driver to do the fast application software development
  - DC-motor

Background Theory

ET-MINI DC-MOTOR

ET-MINI DC-MOTOR module can turn 5V DC motor left or right direction and also can read rotation speed using an optocoupler.

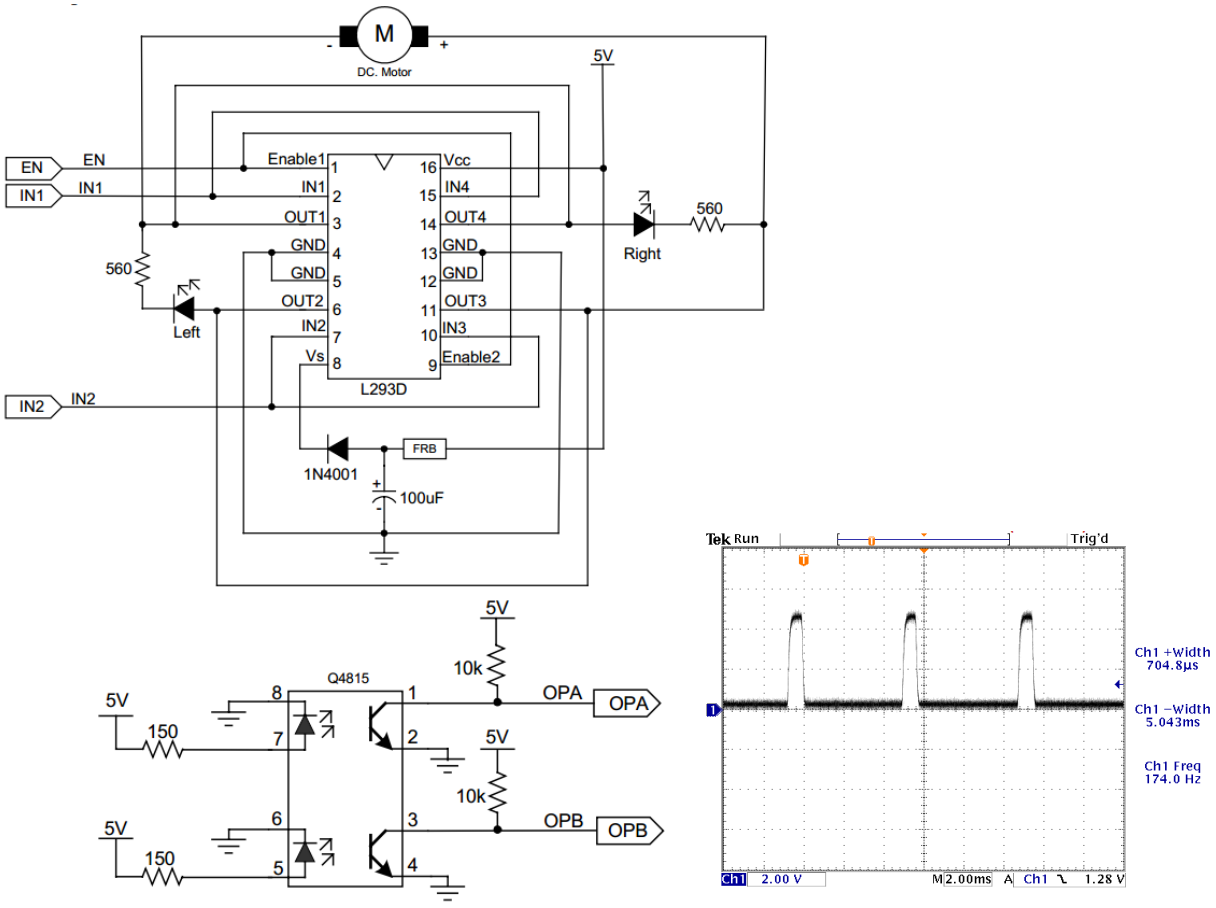
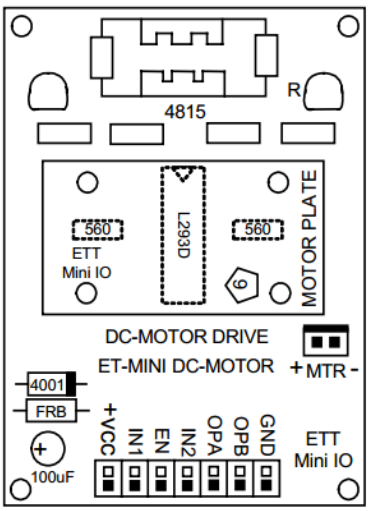


Figure 1 ET-MINI DC-MOTOR circuit



ตารางแสดงการ Control DC-MOTOR

Control DC MOTOR								
RIGHT(R)			LEFT(L)			STOP		
EN	IN1	IN2	EN	IN1	IN2	EN	IN1	IN2
1	1	0	1	0	1	0	x	x

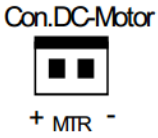


Figure 2 an ET-MINI DC-MOTOR circuit board, pin location and control table

Equipment required

- Nu_LB-002 (Nuvoton learning board)
- ET-MINI DC-MOTOR

Reference:

1. Nu_LB-002 Rev 2.1 User's Manual
2. NuMicro™ NUC130_140 Technical Reference Manual EN V2.02
3. NuMicro™ NUC100 Series Driver Reference Guide V1.05.002
4. L293D datasheet

Procedure 1: ET-MINI DC-MOTOR using Int0

1. Replace the content of the '**Smpl_Start_Kit.c**' with the '**DCmotorInt0**' lab file.
2. Connect the **ET-MINI DC-MOTOR board** with the **Nu_LB-002 learning board**. (Connect 6 wires: **IN1** (short-black-wire) to GPA12 (PWM0), **EN** (short-white-wire) to GPA13, **IN2** (short-red-wire) to GPA14, **Supply** (red-wire) and **GND** (long-black-wire).
3. Connect **OPA** (or OPB) on the **ET-MINI DC-MOTOR board** to **INT0** (GPB14).
4. Compile the project, and run the program.
5. Study the program and work on assignments in the class.

```
uint32_t  Int0Counter = 0;

//------------------------------------------------------------------------Int0
void EINT0Callback(void) {
  Int0Counter++;
  }
//------------------------------------------------------------------------PWM0
void InitPWM0(void) {
  /* Step 1. GPIO initial */
  SYS->GPAMFP.PWM0_AD13 = 1;

  /* Step 2. Enable and Select PWM clock source*/
  SYSCLK->APBCLK.PWM01_EN = 1;  // Enable PWM clock
  SYSCLK->CLKSEL1.PWM01_S = 3;  // Select 22.1184Mhz for PWM clock source

  PWMA->PPR.CP01 = 1;     // Prescaler 0~255, Setting 0 to stop output clock
  PWMA->CSR.CSR0 = 0;     // PWM clock = clock source/(Prescaler + 1)/divider

  /* Step 3. Select PWM Operation mode */
  PWMA->PCR.CH0MOD = 1;   // 0:One-shot mode, 1:Auto-load mode
                //CNR and CMR will be auto-cleared after setting CH0MOD form 0 to 1.
  PWMA->CNR0 = 0xFFFF;
  PWMA->CMR0 = 0xFFFF;

  PWMA->PCR.CH0INV = 0;   // Inverter->0:off, 1:on
  PWMA->PCR.CH0EN = 1;    // PWM function->0:Disable, 1:Enable
  PWMA->POE.PWM0 = 1;     // Output to pin->0:Diasble, 1:Enable
  }
//------------------------------------------------------------------------ADC7
void InitADC7(void) {
  /* Step 1. GPIO initial */
  GPIOA->OFFD |= 0x00800000;  //Disable digital input path
  SYS->GPAMFP.ADC7_SS21_AD6 = 1;    //Set ADC function

  /* Step 2. Enable and Select ADC clock source, and then enable ADC module */
  SYSCLK->CLKSEL1.ADC_S = 2;  //Select 22Mhz for ADC
  SYSCLK->CLKDIV.ADC_N = 1; //ADC clock source = 22Mhz/2 =11Mhz;
  SYSCLK->APBCLK.ADC_EN = 1;  //Enable clock source
  ADC->ADCR.ADEN = 1;     //Enable ADC module

  /* Step 3. Select Operation mode */
  ADC->ADCR.DIFFEN = 0;       //single end input
  ADC->ADCR.ADMD   = 0;       //single mode

  /* Step 4. Select ADC channel */
  ADC->ADCHER.CHEN = 0x80;

  /* Step 5. Enable ADC interrupt */
  //ADC->ADSR.ADF = 1;        //clear the A/D interrupt flags for safe
  //ADC->ADCR.ADIE = 1;
  //NVIC_EnableIRQ(ADC_IRQn);

  /* Step 6. Enable WDT module */
  ADC->ADCR.ADST = 1;
  }
//------------------------------------------------------------------------Timer1
void InitTIMER1(void) {
  /* Step 1. Enable and Select Timer clock source */
  SYSCLK->CLKSEL1.TMR1_S = 0; // Select 12Mhz for Timer0 clock source
  // 0 = 12 MHz, 1 = 32 kHz, 2 = HCLK, 7 = 22.1184 MHz
  SYSCLK->APBCLK.TMR1_EN = 1; // Enable Timer0 clock source

  /* Step 2. Select Operation mode */
  TIMER1->TCSR.MODE = 1;        // 1 -> Select periodic mode
  // 0 = One shot, 1 = Periodic, 2 = Toggle, 3 = continuous counting mode

  /* Step 3. Select Time out period
  = (Period of timer clock input) * (8-bit Prescale + 1) * (24-bit TCMP)*/
  TIMER1->TCSR.PRESCALE = 11; // Set Prescale [0~255]
  TIMER1->TCMPR = 1000000;    // Set TCMPR [0~16777215]
  // (1/12000000)*(11+1)*(1000000)= 1 sec

  /* Step 4. Enable interrupt */
  TIMER1->TCSR.IE = 1;
  TIMER1->TISR.TIF = 1;       // Write 1 to clear the interrupt flag
  NVIC_EnableIRQ(TMR1_IRQn);  // Enable Timer0 Interrupt

  /* Step 5. Enable Timer module */
  TIMER1->TCSR.CRST = 1;      // Reset up counter
  TIMER1->TCSR.CEN = 1;       // Enable Timer0
  }
```

```
109  //-------------------------------------------------------------------Timer1_IRQ
110  void TMR1_IRQHandler(void) {  // Timer0 interrupt subroutine
111    char adc_value[15] = "ADC7 Value:";
112    char int0Count[15] = "INT0 Value:";
113    while (ADC->ADSR.ADF == 0);   // A/D Conversion End Flag
114    // A status flag that indicates the end of A/D conversion.
115
116    ADC->ADSR.ADF = 1;          // This flag can be cleared by writing 1 to self
117
118    PWMA->CMR0 = ADC->ADDR[7].RSLT << 4;
119    sprintf(adc_value+11,"%d   ",ADC->ADDR[7].RSLT);
120    print_lcd(0, adc_value);
121    ADC->ADCR.ADST = 1;          // 1 = Conversion start
122
123    sprintf(int0Count+11, "%d   ", Int0Counter);
124    Int0Counter = 0;
125    print_lcd(1, int0Count);
126
127    TIMER1->TISR.TIF = 1;       // Write 1 to clear the interrupt flag
128    }
129  //-------------------------------------------------------------------------GPIO
130  void InitGPIO() {
131  //  DrvGPIO_Open(E_GPA,12,E_IO_OUTPUT); // IN1
132    DrvGPIO_Open(E_GPA,13,E_IO_OUTPUT); // EN
133    DrvGPIO_Open(E_GPA,14,E_IO_OUTPUT); // IN2
134  //  DrvGPIO_ClrBit(E_GPA,12);
135    DrvGPIO_ClrBit(E_GPA,13);
136    DrvGPIO_ClrBit(E_GPA,14);
137    }
138
139  //-------------------------------------------------------------------------MAIN
140  int32_t main (void) {
141    //Enable 12Mhz and set HCLK->12Mhz
142    UNLOCKREG();
143    SYSCLK->PWRCON.XTL12M_EN = 1;
144    SYSCLK->CLKSEL0.HCLK_S = 0;
145    LOCKREG();
146
147    InitGPIO();
148    // right turn
149    DrvGPIO_SetBit(E_GPA,13); // EN
150    DrvGPIO_ClrBit(E_GPA,14); // IN2
151
152    /* Configure general GPIO interrupt */
153    DrvGPIO_Open(E_GPB, 14, E_IO_INPUT);
154    /* Configure external interrupt */
155    DrvGPIO_EnableEINT0(E_IO_RISING, E_MODE_EDGE, EINT0Callback);
156
157    InitPWM0();       // IN1
158    InitADC7();       // to vary the DCmotor speed
159    InitTIMER1();     // read ADC7
160
161    Initial_pannel(); // call initial panel function
162    clr_all_pannal();
163
164    while (1) {
165      __NOP();
166      }
167    }
```

Figure 3 the program using Interupt0 to count speed

Procedure 2: ET-MINI DC-MOTOR using Timer0 (event counting)

1. Replace the content of the '**Smpl_Start_Kit.c**' with the '**DCmotorTM0'** lab file.
2. Connect the **ET-MINI DC-MOTOR board** with the **Nu_LB-002 learning board**. (Connect 6 wires: **IN1** (short-black-wire) to GPA12 (PWM0), **EN** (short-white-wire) to GPA13, **IN2** (short-red-wire) to GPA14, **Supply** (red-wire) and **GND** (long-black-wire).
3. Connect **OPA** (or OPB) on the **ET-MINI DC-MOTOR board** to **TM0** (Timer0 counter input, GPB8).
4. Compile the project, and run the program.
5. Study the program and work on assignments in the class.

```
//-------------------------------------------------------------------Timer0
void InitTIMER0(void) { // event counting
  /* Step 1. Enable and Select Timer clock source */
  SYSCLK->CLKSEL1.TMR0_S = 2; // Select HCLK for event counting
  // 0 = 12 MHz, 1 = 32 kHz, 2 = HCLK, 7 = 22.1184 MHz
  SYSCLK->APBCLK.TMR0_EN = 1; // Enable Timer clock source

  SYS->GPBMFP.TM0 = 1;  // Multiple Function Pin GPIOB Control Register
  //SYS->ALTMFP.PB9_S11 = 0;  // Alternative Multiple Function Pin Control Register

  TIMER0->TEXCON.TX_PHASE = 1;// A rising edge of external co in will be counted.
  TIMER0->TEXCON.TCDB = 1;    // Enable De-bounce

  TIMER0->TCSR.CTB = 1;       //  Enable counter mode

  /* Step 2. Select Operation mode */
  // TIMER1->TCSR.MODE = 1; // 1 -> Select periodic mode
  // 0 = One shot, 1 = Periodic, 2 = Toggle, 3 = continuous counting mode

  /* Step 3. Select Time out period
  = (Period of timer clock input) * (8-bit Prescale + 1) * (24-bit TCMP)*/
  TIMER0->TCSR.PRESCALE = 0;  // Set Prescale [0~255]
  // TIMER1->TCMPR = 1000000;   // Set TCMPR [0~16777215]
  // (1/12000000)*(11+1)*(1000000)= 1 sec or 1 Hz

  /* Step 5. Enable Timer module */
  TIMER0->TCSR.CRST = 1;      // Reset up counter
  TIMER0->TCSR.CEN = 1;       // Enable Timer

  TIMER0->TCSR.TDR_EN = 1;    // Enable TDR function
  }
```

Figure 4 an Timer0 function as event counting

```
//-------------------------------------------------------------------Timer1_IRQ
void TMR1_IRQHandler(void) {  // Timer0 interrupt subroutine
  char adc_value[15] = "ADC7 Value:";
  char lcd2_buffer[18] = "Timer1:";
  char lcd3_buffer[18] = "T0_TDR:";

  while (ADC->ADSR.ADF == 0); // A/D Conversion End Flag
  // A status flag that indicates the end of A/D conversion.

  ADC->ADSR.ADF = 1;          // This flag can be cleared by writing 1 to self
  PWMA->CMR0 = ADC->ADDR[7].RSLT << 4;
  sprintf(adc_value+11,"%d   ", ADC->ADDR[7].RSLT);
  print_lcd(0, adc_value);
  ADC->ADCR.ADST = 1;         // 1 = Conversion start

  Timer1Counter+=1;
  sprintf(lcd2_buffer+7, " %d", Timer1Counter);
  print_lcd(2, lcd2_buffer);

  sprintf(lcd3_buffer+7, " %d  ", TIMER0->TDR);
  print_lcd(3, lcd3_buffer);
  TIMER0->TCSR.CRST = 1;      // Reset up counter
  TIMER0->TCSR.CEN = 1;       // Enable Timer

  TIMER1->TISR.TIF = 1;       // Write 1 to clear the interrupt flag
  }
```

Figure 5 a modified TMR1_IRQHandler function (every 1 s.) to display speed using Timer0 counter

Assignment(s)

Summarize what you suppose to learn in this class.