Lab 1. DC 모터 PID 제어

실습 0. F/W 컴파일 및 다운로드 테스트

1. 자신의 작업 폴더를 하나 생성. (이하에서는 자신의 작업 폴더를 “D:\USR” 이라고 한다.)
2. 바탕화면에 “C6701” DOS 창 아이콘을 더블 클릭하여 도스 창을 연다. 환경변수가 설정되어야 하므로 반드시 이 도스 창을 활용한다.
3. 과목 홈페이지에서 예제코드(MeTR_PWM.zip)를 다운로드 하여 작업 폴더에 저장하고 압축을 푼다.
4. 도스 창에서 현재 폴더의 위치를 예제코드가 저장된 위치로 이동한다. (cd D:\USR)
5. 도스 창에서 키보드를 통해 “make” 또는 “make ram”을 실행하면 컴파일이 진행되고, “.out” 파일이 생성된다.
6. PC의 USB Port와 실습 KIT를 연결하고, 전원을 인가한다.
7. 도스 창에서 키보드를 통해 “d” 또는 “d.bat”를 실행하면 PC의 USB Port를 통하여 프로그램이 다운로드 되고 실행된다.
8. 하이퍼터미널 등의 serial 통신 프로그램(baud rate: 115,200bps)에서 UART로 전송된 string을 확인할 수 있다.
9. 이 실습에서 사용된 프로그램을 skeleton으로 사용하여 다음 실습을 진행한다. 만일, 위의 동작이 진행되지 않으면 조교에게 문의한다

참고: 위에서 설명한 내용은 DSP의 RAM에 컴파일한 코드를 다운로드하는 절차이다. RAM에 다운로드하였으므로 전원을 껐다 켜면 다운로드한 코드는 사라지고, FLASH에 저장되었던 코드가 실행된다. 만약, 자신이 작성한 코드가 전원이 꺼졌다 켜질 때 자동으로 실행되기를 원하면 실행 코드를 FLASH에 fusing하여야 한다. 이를 위해서는 위의 5번째 단계에서 “make flash”를 실행하여 컴파일하고, 7번째 단계에서는 “f” 또는 “f.bat”를 실행하여야 한다. DSP 보드의 reset 스위치를 누르는 동작은 전원을 껐다 켜는 것과 동일한 효과가 발생한다.

실습 1. PWM을 이용한 DC 모터 구동

1. PWM 발생 회로는 내장 FPGA에 Verilog 코딩으로 구현하였다. 관련 FPGA 레지스터는 다음과 같다.

Register	Bit 31 ~ 1	0
PWMDRVEN (0x02000020)	Reserved to 0's	PWM Driver En 0: Disable 1: Enable

Register	Bit 31 ~ 12	Bit 11 ~ 0
PWMRIGHT (0x02000021)	Reserved to 0's	PWM Duty 0x000 : 0% 0x800 : 50% 0xFFF : 100%

2. 모터의 기구부는 연결하지 않은 상태에서 홈페이지에 배포한 MeTR_PWM.zip 을 실행하여 PWM이 출력되는지 확인.
   • I/O보드 J4의 2번, 3번 핀을 오실로스코프로 관찰하면 50% duty의 서로 역상인 PWM 신호가 관찰됨.
3. *PWMRIGHT에 12 비트의 정수값을 넣었을 때, 위 표와 같이 PWM의 duty가 변화됨을 관찰한다.
4. PWM duty를 설정하는 함수를 다음과 같은 형태로 작성하여 0~100%에 해당하는 PWM 파형이 출력되도록 구현한다.
   void PWMOut(float dutyratio);
   함수의 인자인 dutyratio에는 -50.0 ~ 50.0 범위의 값을 인가하면 PWM 파형은 0~100% 듀티비로 발생하도록 작성한다. (인자에 0을 입력하면 50% 듀티 파형이 생성되어 정지함)
   일단 이 함수를 생성한 후에는 코드에서 PWMRIGHT에 직접 값을 쓰지 말고, 이 함수를 통하여 PWM duty를 변경하도록 한다.
   출력 파형의 PWM duty는 0 또는 100% duty로 saturation 되어야 한다.
5. PWMOut 함수가 동작하면 모터와 보드를 연결하고 PWMOut 함수를 테스트한다.

실습 2. Magnetic 엔코더 신호를 이용한 위치 측정

1. 지금까지의 모터 구동은 PWM의 duty를 조절하는 openloop 구동이다. 이런 경우에는 우리가 원하는 모터의 회전속도로 조절이 용이하지 않으며 모터의 로터를 손으로 잡는 등의 부하가 걸리는 경우에 정속 회전이 불가능하다. 특히, 180도 회전 후 멈추는 등의 위치 제어는 원천적으로 불가능하게 된다.
2. 이를 가능하게 하기 위해서는 모터의 회전을 감지할 수 있는 센서를 사용하여야 한다. 본 실험에서 사용하는 DC 모터에는 모터 1회전 당 512개의 펄스가 발생하는 magnetic 엔코더가 내장되어 있다. 엔코더 신호는 A, B 상으로 두 개가 출력되는데, 90도의 위상차를 가지고 있어서 모터의 회전 방향을 감지할 수 있다. 또한, 모터가 일정 각도가 회전했을 때 구형파가 한 주기씩 발생하므로 구형파의 개수를 카운트하면 모터의 회전 각도를 감지할 수 있다. 마찬가지로 구형파의 주파수는 모터의 회전 속도에 비례하게 된다.
3. 배포한 MeTR_Encoder.zip 을 실행하여 엔코더 카운터 값을 확인한다. *PWMDRVEN 값을 0으로 설정하여 PWM 드라이버를 OFF한 상태에서 *ENCPOSR의 하위 16비트 값을 UART로 대략 0.5초에 1회 display하도록 한 상태에서 바퀴를 손으로 돌려보자. Encoder counter 값이 증가/감소하는 현상을 관찰할 수 있다. 엔코더 파형은 J4의 4번과 5번 핀에서 관찰할 수 있다. 또한, *ENCPOSCLR에 아무 값이나 쓰면 counter의 값이 0으로 reset 된다.

Register	Bit 31 ~ 0
ENCPOSCLR (0x02000023)	아무 값이나 쓰면 position counter 값이 0으로 clear 됨.

Register	Bit 31 ~ 16	Bit 15 ~ 0
ENCPOSR (0x02000083)	Don't Care	Encoder Position Counter 16 bit 2's complement

4. 모터 회전축의 회전 각도를 측정하기 위하여 다음 형태의 함수를 작성한다.

   float GetAngle();

   현재 바퀴의 회전 각도를 deg 단위로 출력하도록 위 함수를 코딩한다.

   모터 회전축 1회전에 엔코더 펄스 카운터 값이 512 증가 또는 감소(회전 방향에 따라)하며, 모터와 바퀴 사이에는 7.5:1 비율의 기어가 장착되어 있음.

   - 모터 1회전 ⇔ 엔코더 펄스 카운터 값 512 증가 또는 감소
   - 바퀴 1회전 ⇔ 모터 7.5회전

실습 3. PID 제어기 구현

1. 홈페이지에 배포한 MeTR_Timer.zip 을 실행하여 TP14에서 500 Hz의 구형파 파형이 출력되는지, 시리얼 통신을 통하여 1초에 한번 스트링이 출력되는지 확인.
2. Timer ISR에서 PID 제어기를 코딩한다. 타이머 인터럽트의 주기는 Constant.h 에서 조정이 가능하다. 현재 인터럽트는 1 kHz로 적용되었다. (특별한 이유가 없는 한 변경할 필요 없음)
3. Timer ISR 내부에서 GetAngle 함수와 PWMOut 함수를 활용하도록 한다.
4. PID gain은 각각 define으로 설정하여 쉽게 수정할 수 있도록 프로그램 한다.
5. position 제어가 제대로 되는지 확인한다. position 제어는 step response를 확인하며, step response를 바탕으로 PID 제어기를 튜닝한다. 이 때, GetAngle을 이용하여 읽은 현재 각도와 reference 각도, 제어 입력 등을 USBMonitor 기능을 활용하여 관측하면 제어공학에서 학습한 오버슛 등의 파형을 확인할 수 있다. USBMonitor 사용법은 일반 오실로스코프 사용법과 거의 같으며, 사용법은 별도 문서 참조.

실습 4. 위치 궤적 트래킹

1. 실습 3에서는 목표 위치를 step 형식으로 적용하였다. 반면, 사다리꼴 속도 프로파일에 대응하는 위치 궤적을 사용하여 부드러운 동작을 구현할 수 있다. 이 위치 궤적은 가속, 등속, 그리고 감속 구간으로 구성되며, 가/감속 구간의 각가속도, 등속 구간의 등각속도, 그리고 이동할 각도 등 3개의 파라미터를 조정하여 궤적의 특성을 변경한다. 이 때, 경우에 따라 등속구간이 나타나지 않는 경우에도 대응하여야 함에 주의한다.
2. 위치 레퍼런스를 생성하는 함수를 생성하고, 이 함수를 main 함수의 while 루프 또는 Timer ISR에서 호출한다.
3. USBMonitor로 각도 레퍼런스, 측정된 각도, 위치 오차, 그리고 제어 입력 등을 확인한다.
4. [선택적] Feedforward 제어기를 적용하면 feedback 제어기의 이득을 높이지 않고도 트래킹 오차를 줄이는데 매우 효과적이다. Feedforward 제어기도 적용하여 적용 전후의 추종 오차를 비교한다.