# Xilinx Zynq FPGA, TI DSP, MCU 프로그래밍 및 회로 설계 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

# Overview

1. 엔코더 구조, 원리, 장단점
2. 리졸버의 구조, 원리, 장단점
3. 활용 분야 및 실제 구현

# Encoder Structure

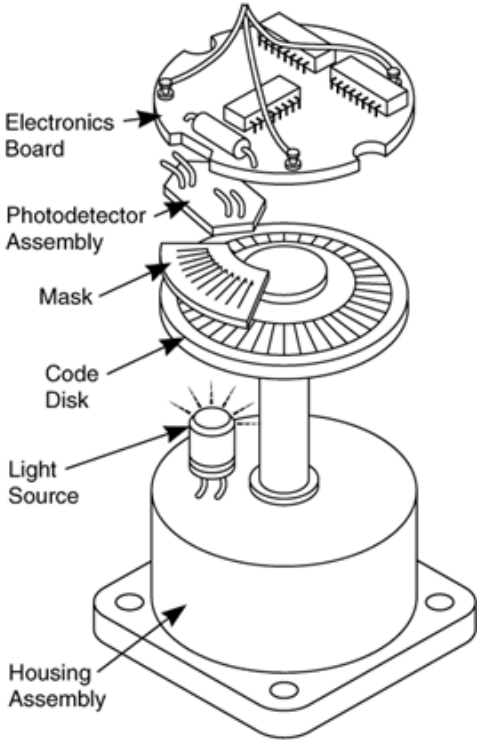
**엔코더의 구조에 대해 알아보도록 하겠다.**

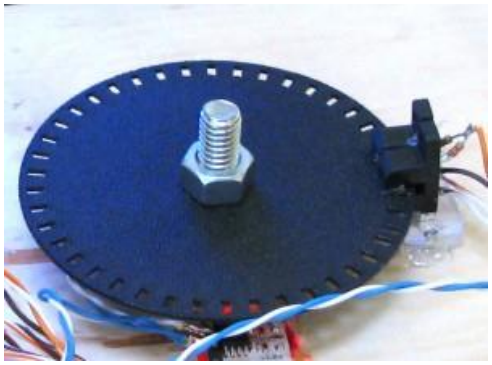
펌웨어 엔지니어가 꼭 정확히 알고 있어야하는 센서중의 하나가 엔코더다.
엔코더는 직류모터와 연결되어 직류모터가 어느 만큼 회전하였는지 회전량을 알려주어
위치 및 속도 제어가 가능하게 해주는 아주 중요한 센서다.

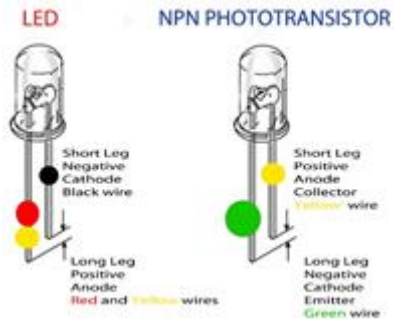
# Incremental Encoder

## 증분 엔코더에 대해 알아보도록 하겠다.

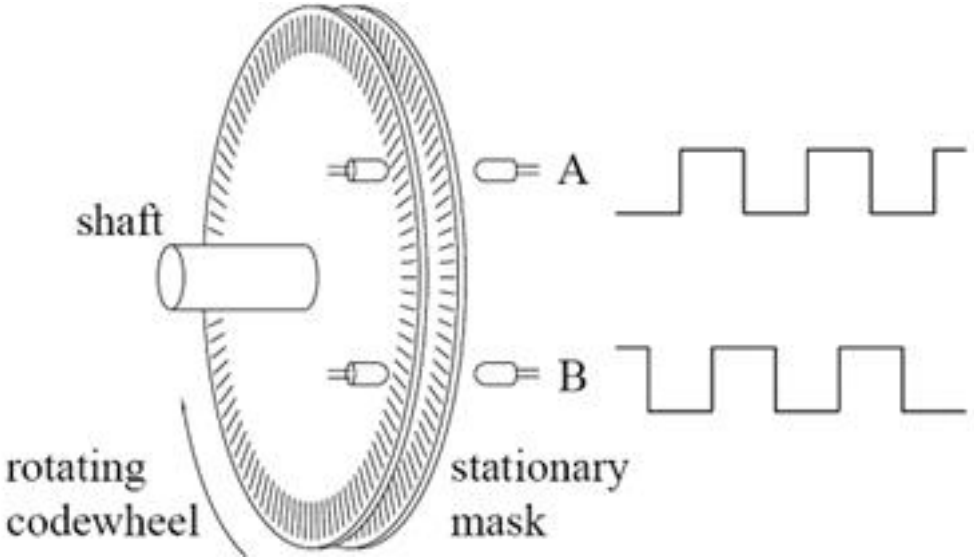
회전하는 부분이 모터와 연결되어 모터와 연동하도록 구성된다.
이 기기의 내부 구조는 아래 그림과 같다.

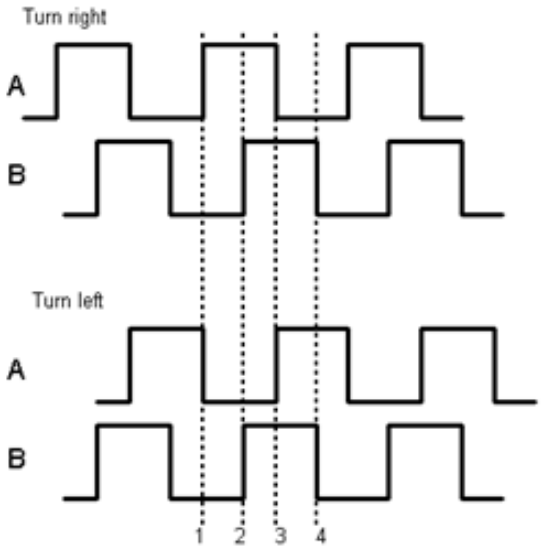





정교하게 제작된 구멍(슬롯)의 양쪽에 빛을 쏘고 받는 장치가 있다(위 그림 오른쪽 아래).
그 발광-수광부 사이를 슬롯이 통과할 때마다 빛은 깜박이게 되고
그 깜박이는 횟수가 결국 슬롯이 있는 원판의 회전량과 같아진다는 원리다.
전체 구성은 위의 왼쪽 그림과 같고, 슬롯은 오른쪽 위 그림과 같다.
그러나 현재의 모양으로는 회전방향을 알 수 없는 문제가 생긴다.




앞선 문제를 해결하기 위해 발광-수광부를 A,B 두 개의 채널로 확장하고
두 개의 슬롯을 약간 차이를 두고 만들게 되면 회전방향(시계방향 또는 반시계방향)에 따라
위의 오른 쪽과 같은 서로 다른 파형이 감지되게 되어 회전 방향을 알 수 있게 된다.

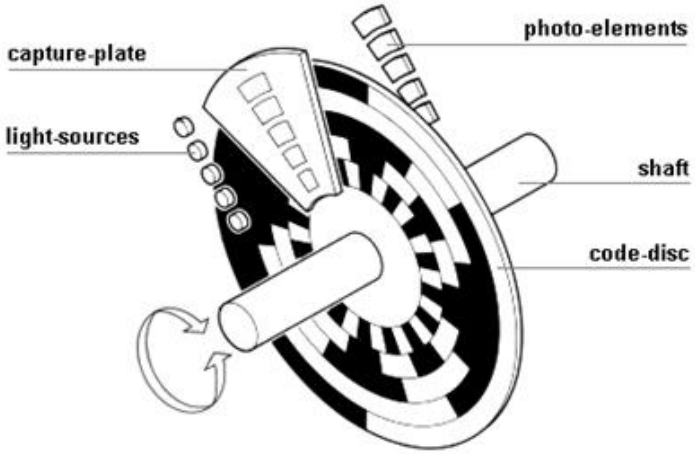
이렇게 하더라도 전원이 새로 켜지면 현재의 모터의 회전각이 얼마인지 알 수가 없다.

# Absolute Encoder

**이번에는 절대 엔코더에 대해 알아보도록 하겠다.**

앞서서 살펴보았듯이 전원이 새로 켜지면 현재의 모터의 회전각이 얼마인지 알 수가 없었다.

그래서 이러한 문제를 해결하기 위해 여러 개의 발광-수광부 짝을 두고
그 센서들이 감지하는 패턴(빛이 통과하고 못통과하는 순서)이 다르도록 만들 필요가 있었으며
그렇게 만들어진 엔코더를 절대 엔코더라 하고 앞서 설명한 것은 상대 엔코더라 부른다.





실제 상용화된 절대엔코터의 원판 모양은 다음 그림처럼 아주 복잡하다.
하지만 오른쪽 그림의 금고 다이얼이라면 복잡하더라도 언제나 지금 위치를 알 수 있어야 한다.

# Speed Measurements(M Method)

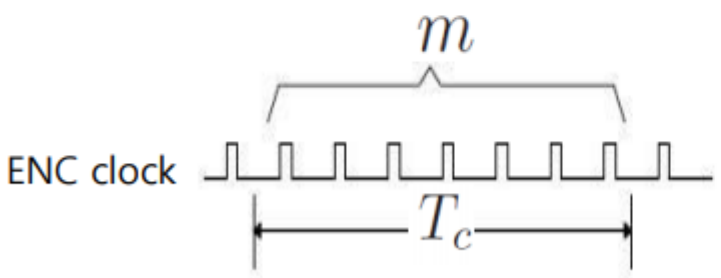
**이번에 속도를 측정하기 위한 M 방식을 알아보도록 하겠다.**

일정 시간마다 펄스를 세는 방법으로 예로 1 초(T)동안 펄스가 몇 개 들어오는지 카운트(X)를 하여 계산한다.
예로 펄스를 1 초마다 세는 펄스가 3 초에 한 번씩 들어오는 저속의 경우 측정이 이뤄지지 못한다.
즉 M 방식은 고속으로 모터가 회전할 때 적합한 방식이다.

## ① M 방식

- 구현 간단
- PPR(회전당 펄스 수) 작으면 정밀도 떨어짐
- 저속에서 정확성 떨어짐

| 각속도 | 분당회전수(RPM) | 변위 |
|---|---|---|
| $\omega_m = \frac{X}{T}$ | $N = \frac{60}{2\pi}\frac{X}{T}$ | $X = \frac{m}{PPR}2\pi$ |




$$\omega_m = \frac{X}{T_c} = \frac{2\pi}{T_c}\frac{m}{PPR}$$

$$N = \frac{60}{2\pi}\omega_m = \frac{60}{T_c}\frac{m}{PPR}$$

# Speed Measurements(T Method)

**이번에 속도를 측정하기 위한 T 방식을 알아보도록 하겠다.**

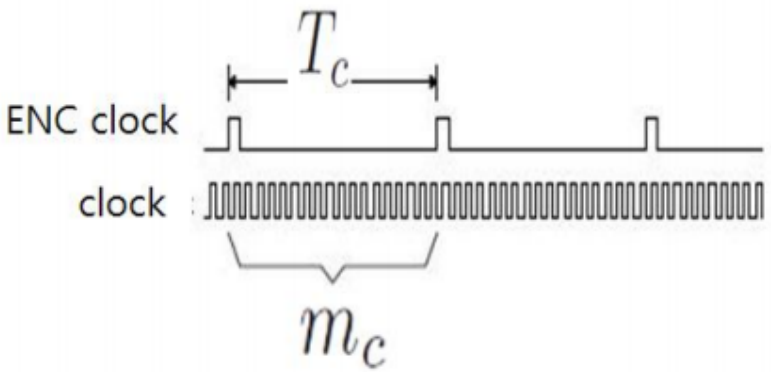
이 방법은 모터에서 들어오는 펄스 사이의 간격을 측정하는 것이다.
즉 펄스가 들어오는 간격인 $T_c$ 를 $m_c$ (MCU 의 카운터) 펄스의 개수로 측정하는 것이다.
MCU 관점에서 하나의 펄스가 들어올때 카운터를 증가시켜서 다음 펄스가 들어올때 다음 카운터를 읽으면 된다.
M 방식과는 반대로 고속일 경우 문제가 되는데 펄스가 무척 빠르므로 카운터가 1 개 증가하는 동안
수십개의 모터 펄스가 들어온다면 속도를 정확히 파악할 수가 없다.
또한 무작정 카운터를 빠르게 증가하게 만든다면 저속에서 펄스 간격이 길어져 타이머 오버플로우가 발생하게 된다.

## ② T 방식

- 저속에서 정밀한 속도 측정 가능
- 정확도 위해 고속영역에서 클록 주파수 증가
  → 저속 영역에서 계수해야 할 mc 증가
- Tc 가 가변 → 일정 샘플링 주기의 속도 제어기에 부적합

$$X = \frac{2\pi}{PPR}$$

$$T_c = m_c T_{clock}$$




$$\omega_m = \frac{X}{T} = \frac{2\pi}{PPR}\frac{1}{m_c T_{clock}}$$

$$N = \frac{60}{2\pi}\omega_m = \frac{60}{PPR}\frac{1}{m_c T_{clock}}$$

# Speed Measurements(M/T Method)

**이번에 속도를 측정하기 위한 M/T 방식을 알아보도록 하겠다.**

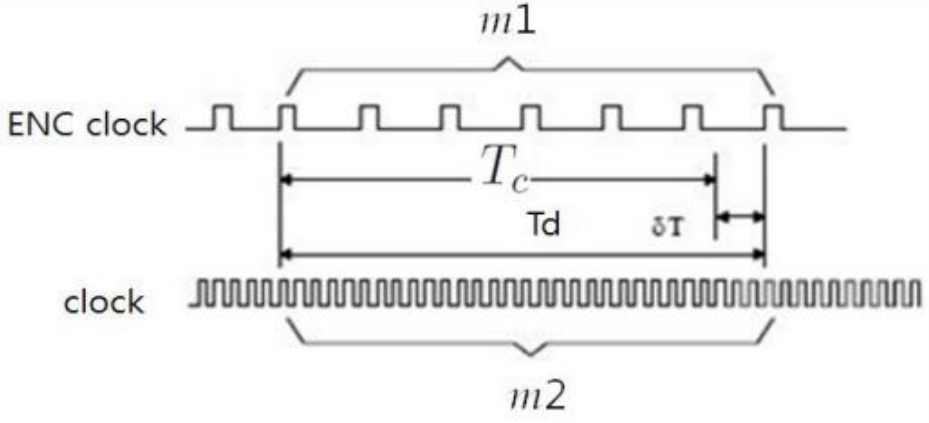
이 방법은 M 방법에 기반을 두고 T 방법을 추가하는 것과 비슷한 방법으로 카운터를 2 개 사용한다.
M 방법을 주파수가 낮은 카운터로 펄스의 개수를 카운트($T_c$) 한다.
그리고 주파수가 높은 카운터로 펄스의 개수를 샌 순간부터 다음 모터의 펄스가 들어오는 시간($\delta T$)을 측정한다.
$T_d = T_c + \delta T$ 로 이 정보를 기반으로 정확한 속도를 측정할 수 있다.

또 다른 방법으로 펄스의 시간 간격을 측정하되 고속 타이머와 저속 타이머를 동시에 동작시켜
시간 간격이 짧은 경우 고속 타이머로 측정한 시간을 사용하고
긴 경우엔 저속 타이머로 측정한 시간을 활용한다면 보다 효율적으로 속도를 측정할 수 있다.

## ③ M/T 방식

- 구현 복잡
- 극저속에서 $\Delta T$ 가 커지게 되면 속도 측정 주기가 변동하게 되는 단점을 여전히 가짐

$$X = \frac{2\pi}{PPR} m_1$$




$$T = T_c + \Delta T = T_c + m_2 T$$

$$\omega_m = \frac{X}{T} = \frac{2\pi}{PPR} \frac{m_1}{T_c + m_2 T_2}$$

$$N = \frac{60}{2\pi} \omega_m = \frac{60}{PPR} \frac{m_1}{T_c + m_2 T_2}$$

# Resolver Structure

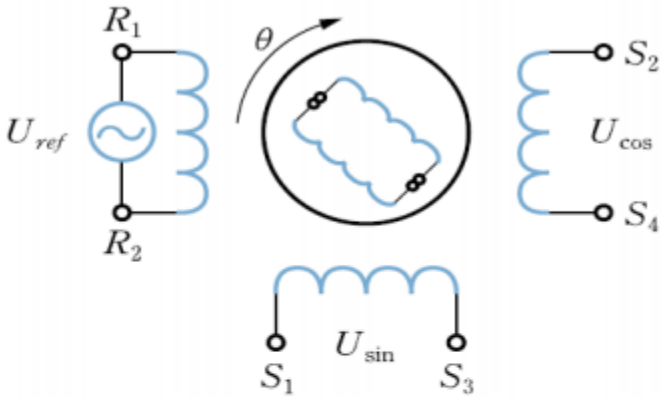
## 리졸버의 구조에 대해 알아보도록 하겠다.

위치 감지 기술로 잘 알려진 리졸버(resolver)는 변압기(transformer)와 어느 정도 비슷하게 동작한다.
리졸버는 스테이터와 로터로 이루어졌으며 로터는 모터 샤프트(shaft)에 고정된다.
또한, 어떠한 전자 장치도 내장되어있지 않기 때문에 먼지, 고온, 고속 등의 혹독한 환경에 적합하다.

이 역시 트랜스포머와 흡사하게 리졸버의 스테이터 권선이 일차 측에서
여자 전압을 수신한 후 로터 권선이 이차 측에서 전자기 결합을 통해 전압을 발생시킨다.
아래 그림을 보면 출력 전압 진폭은 로터 각도 변위와 상관적인 사인-코사인을 나타낸다.
각도 변위는 출력 전압을 디지털로 변환하고 역 탄젠트(arc-tangent)를 계산해서 구할 수 있다.

# ① 리졸버의 구조

- 인코더보다 충격이나 진동, 온동 변화에 견실하며, 소형화가 가능
- 회전자의 위치를 계산해주는 별도의 소자(RDC)와 여자시키는 회로가 필요
- 회전자의 불균일성 또는 여자 회로로 인한 속도상의 오차 발생 가능성이 있음
- 리졸버는 인코더보다 값이 비싼 편임




$$U_{ref} = U_{amp}\sin(w_{ref}t)$$

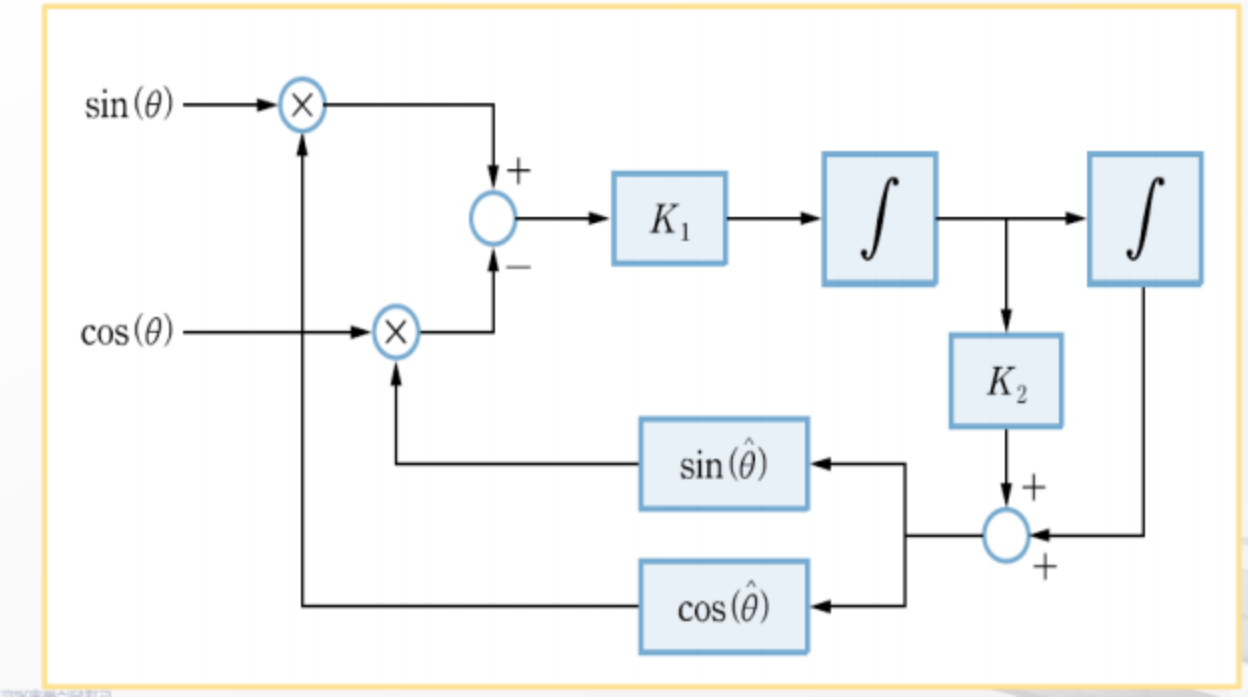
$$U_{\sin} = K\,U_{ref}\sin(\theta)$$

$$U_{\cos} = K\,U_{ref}\cos(\theta)$$

벡터 제어가 중요한 AC Motor 에서는
현재 회전자의 위치를 정확하게 예측할 필요성이 있기에
아래와 같은 알고리즘 또한 매우매우 중요하다.

## ② 회전자 위치 추정 알고리즘

- 블록 다이어그램

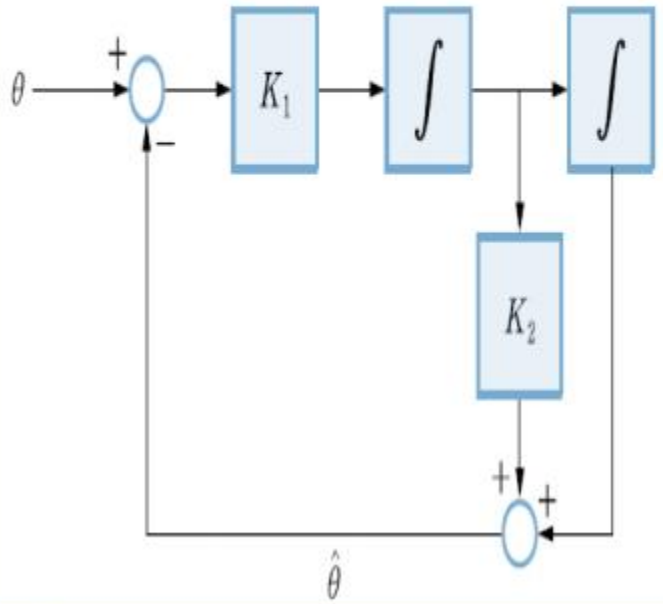


수능을 봤던 학생들은 샌드위치 정리라는 것으로 기억할 가능성이 높겠지만
우리 같은 공학도들에겐 테일러 급수 혹은 매클로닌 급수로 더 유명한 녀석이다.

사실 물리학에서 원운동에 대한 근사식을 적거나 전자기학에서도 많이 사용하는 기법이 테일러 급수인데
sin 의 각도를 limit 0 에 근사하게 가져갈때 0 으로 수렴하는 것이 아니라
각도 자체로 수렴한다는 것을 활용하여 아래와 같은 식을 만들어낸다.

## ② 회전자 위치 추정 알고리즘

- 선형화 모델과 전달함수




$$\hat{\theta}(t) = K_2 \int_0^t K_1 \left( \theta(\tau) - \hat{\theta}(\tau) \right) d\tau + \int_0^t \int_0^\tau K_1 (\theta(r) - \hat{\theta}(r)) dr d\tau$$

$$\hat{\theta}(s) = \frac{K_2 K_1}{s} (\theta(s) - \hat{\theta}(s)) + \frac{K_1}{s} \mathcal{L} \left( \int_0^t (\theta(r) - \hat{\theta}(r)) dr \right)$$
$$= \frac{K_2 K_1}{s} (\theta(s) - \hat{\theta}(s)) + \frac{K_1}{s^2} (\theta(s) - \hat{\theta}(s))$$

**최종적인 전달함수**

$$\frac{\hat{\theta}(s)}{\theta(s)} = \frac{K_1 (1 + K_2 s)}{s^2 + K_1 K_2 s + K_1}$$

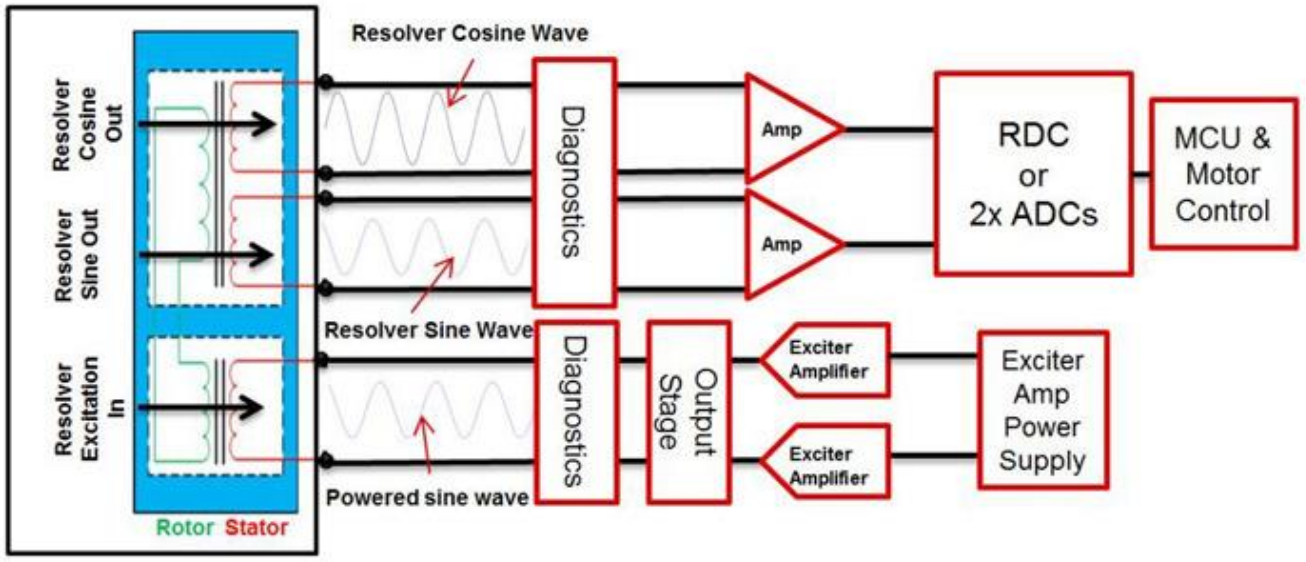
# Resolver Topology Peripheral Circuit

## 리졸버 센서 주변의 다수의 부가 기능을 가진 주변 회로를 살펴보겠다.

아래에 보이는 것과 같이 리졸버 센서 주변으로 다수의 기능들을 필요로 한다.

리졸버를 구동하기 위해서 센서는 전력 스테이지와 여자 증폭기가 필요하다.
전력 스테이지는 여자 증폭기에 전력을 공급하며, 종종 센서 자체로 전력을 공급하기도 한다.
여자 증폭기는 리졸버 센서가 사인 및 코사인 값을 발생시키기 위해 필요한 입력 사인파를 발생시킨다.

리졸버로의 출력은 아날로그 프론트 엔드(AFE)를 필요로 하는 아날로그 신호로
각도를 계산 하는 마이크로컨트롤러나 제어 장치의 신호를 정리하고 조정한다.
리졸버-디지털 컨버터(RDC)는 사인파 및 코사인파를 처리할 수 있도록 디지털 도메인으로 변환한다.
이들 출력이 SPI(serial peripheral interface), 병렬, 에뮬레이트 엔코더
및 아날로그 같은 다양한 옵션을 통해서 각도 및 속도 정보를 MCU로 전달할 수 있다.
이 시스템은 또한 과전류, 과전압, 단락 회로, 과열 보호 같은 센서와
전기 장치들 간의 다중 보호 및 결함 진단 기능의 이점을 제공한다.



# Where we can use it ?

## 이를 사용하기에 적합한 응용단에는 무엇이 있을지 알아보겠다.

앞서 언급했듯이 리졸버 내부에는 어떠한 전기 부품이 없으므로
이 센서는 고온, 먼지, 고속(8,000 rpm), 심한 진동과 같은 조건에서 문제 없이 작동한다.
또한 리졸버는 이렇게 전기 부품들이 없기에 다른 센서 보다 수명이 길다.

아래 보이는 것과 같이 리졸버는 산업용 및 자동차 애플리케이션에 적합하다.
서보 제어 시스템(엘리베이터), 산업용 로봇, AC 인버터 드라이브,
플라스틱 압축 시스템, 회전 시스템, 금속가공 시스템에 리졸버를 사용할 수 있다.
자동차 분야에서는 하이브리드 전기차/전기차(HEV/EV) 트랙션 인버터, 히팅,
HVAC 시스템, 스톱-스타트 얼터네이터, 파워 스티어링 시스템등에 사용할 수 있다.

# Actual Using Encoder

## 실제 프로젝트에 사용해본 엔코더에 대해 파악을 해보도록 하겠다.

실제 프로젝트에 사용할 모터를 선정하기 위한 방법을 살펴보도록 하겠다.
프로젝트에서 모터의 속도를 측정하기 위해 오토닉스 축형 증분 로터리 엔코더를 사용하였다.
데이터시트를 보면 다음과 같은 정보가 나와있다.

$$\left[\text{최대응답회전수 (rpm)} = \frac{\text{최대응답주파수}}{\text{분해능}} \times 60\ \text{sec}\right]$$

[PLC 조무이 가능하니다]

최대허용회전수 ≥ 최대응답회전수 조건이 되도록 분해능을 선정

외경 Ø50
펄스수 1000~8000

위 정보를 기반으로 해당 엔코더가 커버칠 수 있는 RPM 을 산출할 수 있다.
즉 사용하는 모터가 RPM 을 아득히 뛰어넘을 경우에
해당 엔코더로는 속도를 측정할 수 없게 된다.

여기서 또 고려해야할 것이 토크와 속도와의 관계다.
정지한 물체가 이동을 하기 위해서는 정지 마찰력을 이겨내야 한다.
반면 한 번 이동한 물체는 이동 마찰력만 이겨내면 된다.
(보편적으로 정지 마찰력이 이동 마찰력보다 강하다)

즉 이를 이겨내기 위해서는 모터의 토크가 강해야한다.
문제는 일반적으로 모터 혼자 내놓을 수 있는 토크는 약하다는 것이다.

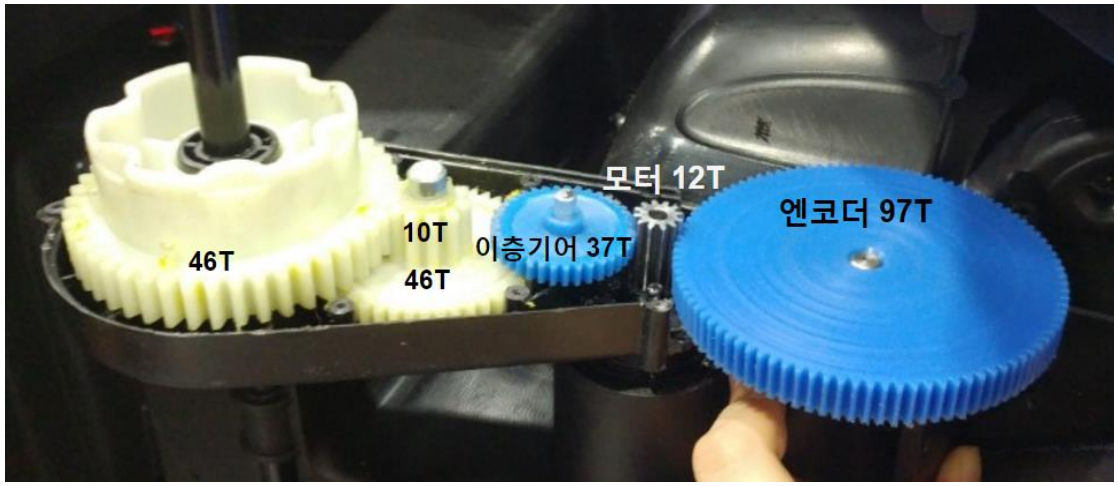
**실제로 아래와 같은 유아용 자동차를 개조하는 프로젝트를 진행했습니다.**


| | | | |
|---|---|---|---|
| 사용연령 | 만3~8세 | 최대적재량 | 30kg |
| 배터리사양 | 12V 7.0 Ah | 차체중량 | 18kg |
| 충전시간 | 8시간 | 적용모터 | 12V 듀얼모터 |
| 사용시간 | 약 1시간 30분 | 속도 | 2.5~5km |
| 제품크기 | 118*69.5*52cm | 충전기전원 | AC 240V 60Hz / 15V 1A |

어린이안전확인인증번호: CB064R1079-7001

문제는 엔코더를 연결해서 Feedback 제어를 수행하려고 했더니
프레임 자체가 DC 모터로 제어하는 구조이며 속도 제어를 전혀 고려하지 않은 구조였다.
이와 같은 이유로 DC 모터 자리에 BLDC 모터를 끼우고 속도 제어를 수행하기 위해 엔코더를 함께 달았다.

이를 수행하기 위해 프레임에 구멍을 뚫고 별도로 엔코더를 장착할 수 있게 만들었으며
엔코더가 모터의 회전하는 기어와 맞물려서 함께 회전할 수 있도록 별도의 기어 제작이 필요했다.
그로 인해 아래와 같은 파랑색의 새로운 기어를 제작하게 된다.




12:37, 37:46, 10:46 을 고려한다.
먼저 모터에 연결된 기어비는 3.0833333333 으로 3.083333 정도에 해당한다.
다음으로 이층 기어인 37 아래에 10 과 물린 46 의 기어비는 4.6 이다.
다음은 10 과 물린 46 으로 기어비로 4.6 이다.
즉 모터가 회전하면서 최종적으로 바퀴에 전달하게 되는 기어비는 65.2433 이다.
즉 6.5 kg 을 1 cm 움직이게 하던 토크가 기어를 통해 증폭되면
424.08 kg 을 1 cm 움직이게 할 수 있는 아주 강력한 토크를 만들어내게 된다.
물론 기어가 물려서 토크가 증가하는 만큼 속도는 그만큼 떨어지게 되어있다.

모터와 엔코더의 비는 8.08333 으로 모터가 빠른 속도를 낼지라도 대략 8 배 가량 그 속도가 감쇠되어 측정되게 된다.
그러므로 엔코더에서는 기어비를 감안하여 실제 속도를 계산해주면 된다.
마찰등의 오차가 존재할 수 있으므로 가속도 센서와 비교해 오차를 보정해주는 것도 좋은 방법이 될 수 있다.

여기에 카메라 밑 레이더등을 추가 장착하여 유아용 자율 주행 차량을 만들 수 있다.
레이더의 경우에는 데이터량이 많아 최소 DSP 나 FPGA 로 연산을 수행하는 것이 정보를 처리하는데 효과적이다.
즉 일반적으로 Firmware 를 제어하는데 사용하는 Cortex-M 이나 Cortex-R 과 달리
Application 용 프로세서인 Cortex-A 가 필요하다.

이런 부분들을 적절하게 조합하여 아래와 같은 결과물을 만들 수 있었다.
참고로 사용한 레이더는 77 GHz Long Range Radar 를 사용하였다.
좀 더 Wide 한 Short Range Radar 도 사용하여
현존하는 ADAS System 처럼 만들고도 싶었으나 금전이 딸려 이정도에서 마무리하였다.

참고로 사용한 DSP 는 TI 사의 TSM320C6678 이며 FPGA 는 Zynq Zybo 보드를 사용하였다.

# How to use Encoder with MCU

**TMS570LC4357 의 데이터시트에 eQEP 라는 부분이 존재한다.**

해당 부분이 Cortex-R5F MCU 에서 Encoder 를 서포트하는 부분에 해당하며
아래 빨간 박스로 표시된 것은 PINMUX 로
경우에 따라 기능을 바꿀 수 있는 핀들이다.
즉 MCU 차원에서 PINMUX 를 제어해서 eQEP 를 활성화시킨다.

| | | | | | |
|---|---|---|---|---|---|
| 254h | ECAP4 | N/A on 337ZWT | PINMMR81[0] | G19 | PINMMR81[1] |
| | ECAP5 | N/A on 337ZWT | PINMMR81[8] | H18 | PINMMR81[9] |
| | ECAP6 | N/A on 337ZWT | PINMMR81[16] | R2 | PINMMR81[17] |
| | eQEP1A | N/A on 337ZWT | PINMMR81[24] | V9 | PINMMR81[25] |
| 258h | eQEP1B | N/A on 337ZWT | PINMMR82[0] | W9 | PINMMR82[1] |
| | eQEP1I | N/A on 337ZWT | PINMMR82[8] | V10 | PINMMR82[9] |

A = V9
B = W9
Z = V10

◎ 후면 배선인출 커넥터형 / 후면/측면 커넥터형

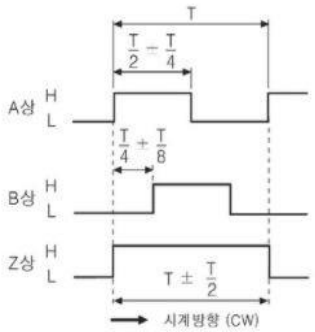
● Totem pole 출력
NPN 오픈 콜렉터 출력
전압출력

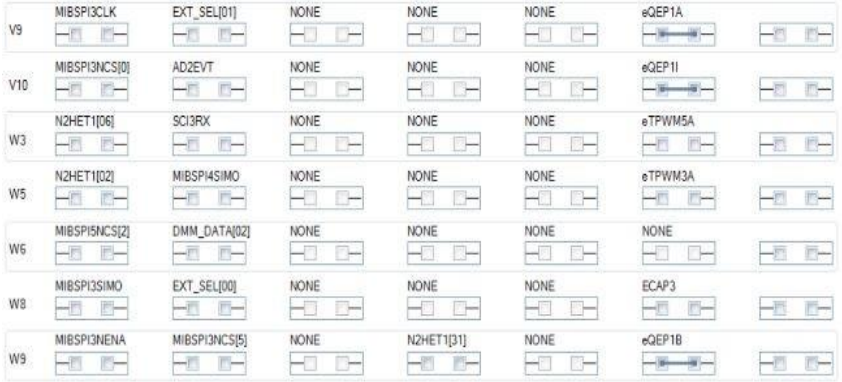
● Line driver 출력

| • Totem pole 출력<br>• NPN 오픈 콜렉터 출력<br>• 전압출력 | | | • Line driver 출력 | | |
|---|---|---|---|---|---|
| 핀 번호 | 기능 | 배선색상 | 핀 번호 | 기능 | 배선색상 |
| 1 | OUT A | 흑 | 1 | OUT A | 흑 |
| 2 | OUT B | 백 | 2 | OUT $\bar{A}$ | 적 |
| 3 | OUT Z | 등 | 3 | +V | 갈 |
| 4 | +V | 갈 | 4 | GND | 청 |
| 5 | GND | 청 | 5 | OUT B | 백 |
| 6 | F.G. | Shield | 6 | OUT $\bar{B}$ | 회 |
| — | | | 7 | OUT Z | 등 |
| | | | 8 | OUT $\bar{Z}$ | 황 |
| | | | 9 | F.G. | Shield |

▣ 출력 파형

● Totem pole 출력 /
NPN 오픈 콜렉터 출력 / 전압 출력



# Important Register on eQEP

**데이터시트 내에서 중요한 eQEP 레지스터들을 몇 가지 살펴보겠다.**

**아래와 같은 레지스터들을 이용하여 앞서 살펴본 속도 측정 기법으로 현재의 속도를 계측할 수 있다.**

Table 33-4. eQEP Position Counter Register (QPOSCNT) Field Descriptions

| Bits | Name | Description |
|---|---|---|
| 31-0 | QPOSCNT | This 32-bit position counter register counts up/down on every eQEP pulse based on direction input. This counter acts as a position integrator whose count value is proportional to position from a give reference point. |

**Pulse** 개수를 카운트

Table 33-11. eQEP Unit Timer Register (QUTMR) Field Descriptions

| Bits | Name | Description |
|---|---|---|
| 31-0 | QUTMR | This register acts as time base for unit time event generation. When this timer value matches with unit time period value, unit time event is generated. |

**VCLK3(**기본 **75MHz)** 를 이용하는 **Unit Timer**

Table 33-12. eQEP Unit Period Register (QUPRD) Field Descriptions

| Bits | Name | Description |
|---|---|---|
| 31-0 | QUPRD | This register contains the period count for unit timer to generate periodic unit time events to latch the eQEP position information at periodic interval and optionally to generate interrupt. |

**QUTMR**의 주기 **(MAX**값**)**을 정해줌 **QUTMR = QUPRD**가 되면 **Unit time event**발생

Table 33-10. eQEP Position Counter Latch Register (QPOSLAT) Field Descriptions

| Bits | Name | Description |
|---|---|---|
| 31-0 | QPOSLAT | The position-counter value is latched into this register on unit time out event. |

**Unit time event** 가 발생하면 **QPOSCNT**의 값을 저장

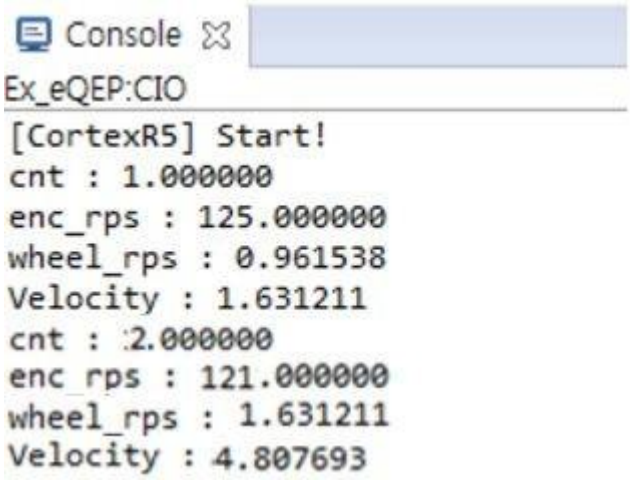
# Firmware Implementation

**펌웨어 단에서 구현한 코드는 아래와 같다.**

**속도는 km/h 로 환산하여 velocity 변수에 저장한다.**
**코드상에 보면 해상도, 펄스개수, 앞서 계산한 기어비등이 모두 활용됨을 볼 수 있다.**
**각 변수들을 매크로화 시켜서 해상도, 펄스수, 기어비 변동에 유연하게 대처할 수 있게 구현하였다.**
**참고로 wheel_r 변수는 실제 프레임의 바퀴 반지름에 해당한다.**

```
#define PI 3.141592
#define wheel_r 0.000075 //km
#define gear_ratio 130
#define resolution 2000
#define pulse_per_sec 250000
```

```
double cnt = eqepREG1 -> QPOSLAT;
double enc_rps = pulse_per_sec / resoulution * cnt;
double wheel_rps = enc_rps / gear_ratio;
double velocity = 2 * PI * wheel_r *3600 * wheel_rps;
```



# Actual Using Resolver

## 실제 사용해본 리졸버에 대해 파악을 해보도록 하겠다.

일전의 프로젝트를 수행해보기 전까진 AC Motor 들이 내부에 리졸버를 가지고 있는지 전혀 몰랐었다.
특히 C2000 계열에서 모터 제어를 수행해본것은 단순한 호기심으로
AC 모터와 일반 DC 모터는 파형이 다르니 맥스웰 방정식의 패러데이 법칙과 암페어 법칙을 기반으로
코일로 구성된 모터에 파형이 떳다가 내려가는 도중에도 토크를 유도할 수 있는 제어식에 대해 학습해보고 싶었다.


단순히 이와 같은 이유로 그림에 보이는 TI 사의 C2000 계열 평가 보드를 구했고
그 옆에 무식하게 커다란 ACIM(AC Induction Motor)를 구매하였다.

현재 그림에 보면 커다란 콘덴서 주변에 선 3 개가 연결되어 있는 것을 볼 수 있는데
이 3 개의 선이 실제 ACIM 모터에 입력되는 전기 신호를 제공하는 3 개의 선이다.
또 살펴보면 모터의 앞대가리 근처에 또 다른 선이 보이는데
이 선이 또 보드의 다른곳에 연결되어 있는 것을 볼 수 있다.
해당 부분이 ADC 로 모터에서 출력된 정보가 보드로 피드백되는 구간이다.
바로 이 선이 리졸버의 정보가 나오는 선이였다.

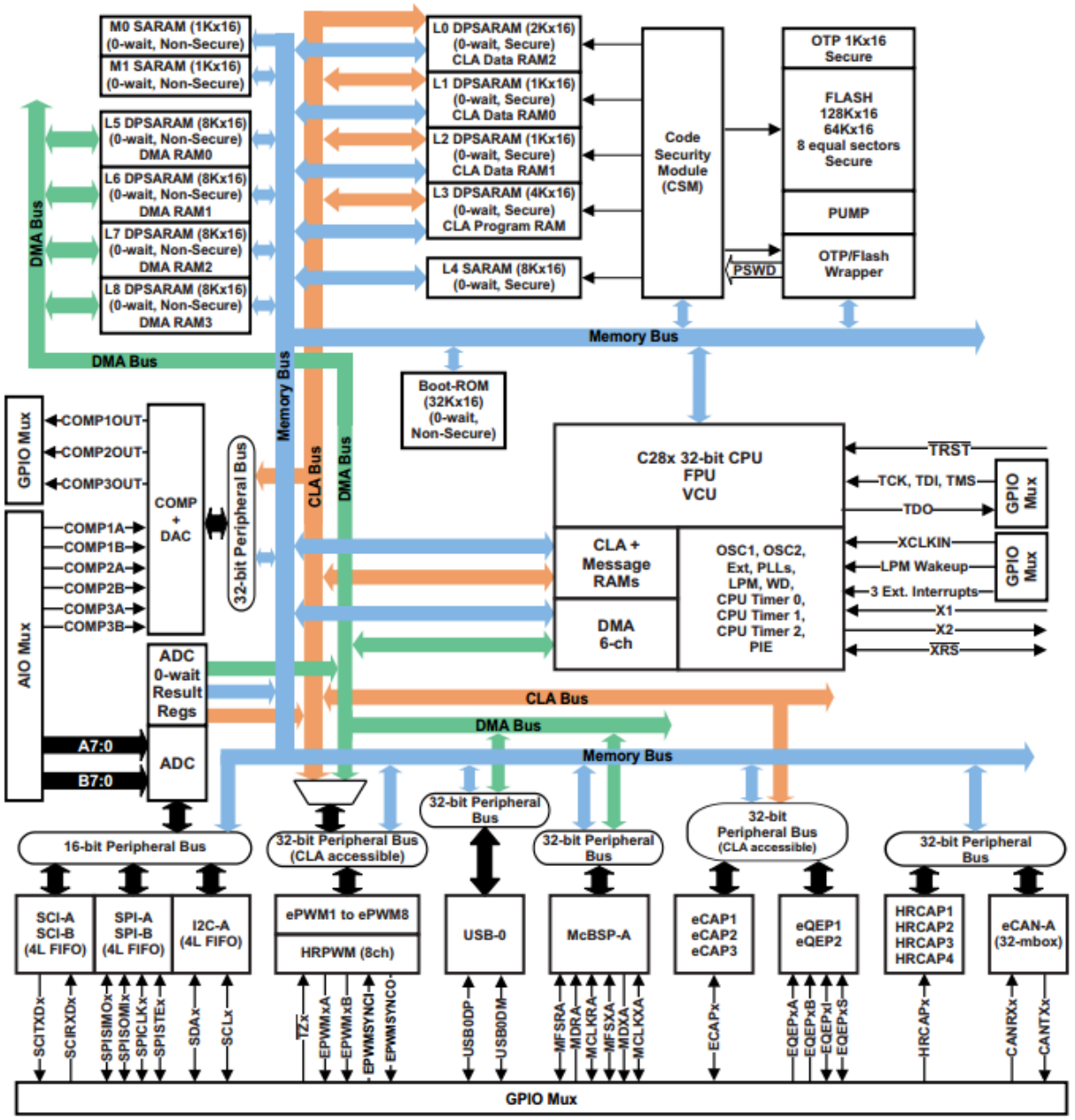
이 녀석을 가지고 무언가 프로젝트를 만들어보진 않았다.
당시 단순한 학습용 목적으로 진행했던것이지만
정리하면서 이 부분이 리졸버였다는 것을 알 수 있었다.

동일한 솔루션을 구해서 ACIM 모터에 대한 학습을 진행하고자 한다면
아래 링크를 활용해볼 수 있다.

http://www.ti.com/tool/tmdshvmtrinspin?keyMatch=High%20Voltage&tisearch=Search-EN-Everything

## 1.4 Functional Block Diagram

Figure 1-1 shows a functional block diagram of the device.




A. Not all peripheral pins are available at the same time due to multiplexing.

**Figure 1-1. Functional Block Diagram**

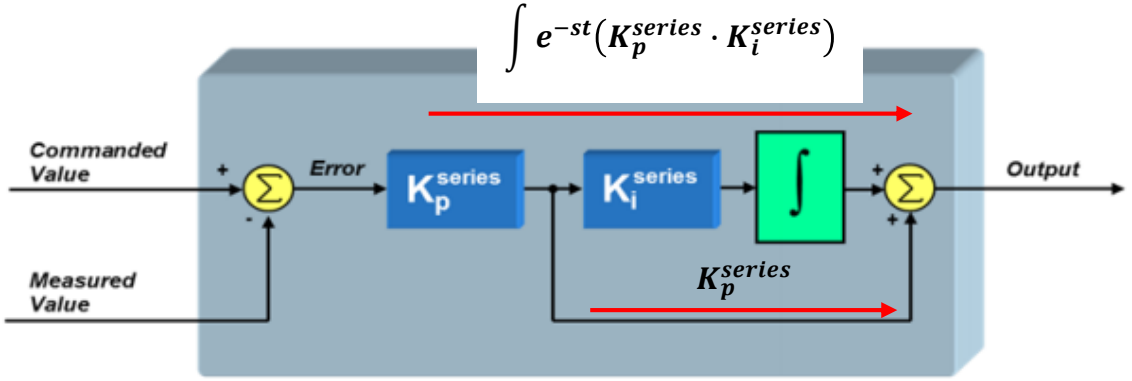
# How to analysis Block Diagram ?

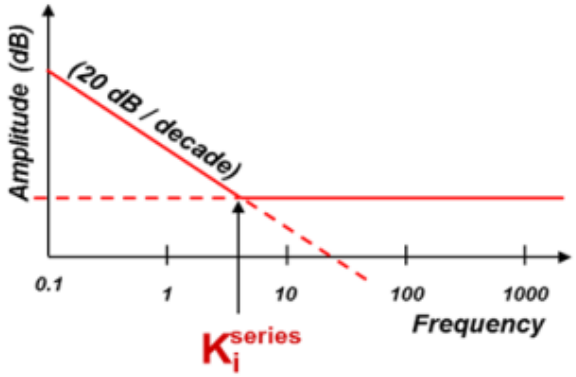
**C2000 의 경우 TI 사에서 제공하는 문서에 블록 다이어그램을 해석하는 방법이 있다.**

다만 중간 과정들이 생략된 것들이 많아 쉽게 이해하기 어려운 구조로 구성되어 있다.

**Background** 그래서 아래와 같이 이해를 쉽게 하기 위해 라플라스 변환을 적용한 결과를 입력해봤다.

A popular form of the PI controller (and the one used for this analysis) is the "series" topology which is shown below.




$K_p^{series} = K_p$

$K_i^{series} = K_i / K_p$

$K_p^{series}$ *is simply a series gain term*

$K_i^{series}$ *term specifies the inflection frequency*

Figure 17: Series PI control.

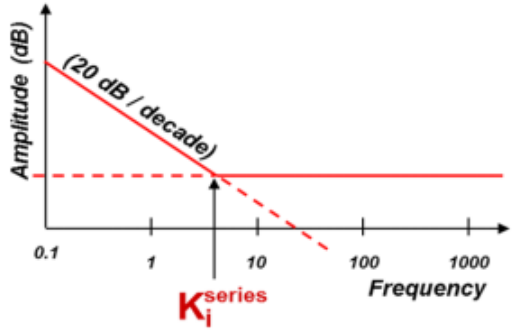
From this diagram, we can see that:

$$K_p^{series} = K_p$$
$$K_i^{series} = \frac{K_i}{K_p}$$

Where $K_p$ and $K_i$ are the parallel PI controller proportional and integral gains respectively. In the series structure, $K_p^{series}$ sets the gain for ALL frequencies, and $K_i^{series}$ directly defines the inflection point (zero) of the controller in rad/sec. It's pretty easy to understand the effect that $K_p^{series}$ has on the controller's performance, since it is simply a gain term in the open-loop transfer function. But what is the system significance of the zero inflection point?

It is common knowledge that the gain of the PI controller has a pronounced effect on system stability. But it turns out that the inflection point in the graph (the "zero" frequency) also plays a significant but perhaps more subtle role in the performance of the system. To understand this, we will need to derive the transfer function for the PI controller, and understand how the controller's "zero" plays a role in the overall system response.

다음으로 앞 페이지에서 정리한 수식을 기반으로 이 부분을 고스란히 진행할 수 있다.




Using the series form of the PI controller, we can define its "s-domain" transfer function from the error signal to the controller output as:

$$PI(s) = \frac{K_p^{series} \cdot K_i^{series}}{s} + K_p^{series} = \frac{K_p^{series} \cdot K_i^{series}\left(1 + \frac{s}{K_i^{series}}\right)}{s}$$

Equation 1

From this expression, we can clearly see the pole at s = 0, as well as the zero at s = $K_i^{series}$ (rad/sec). So, why is the value of this zero so important? To answer this question, let's drop the PI controller into the heart of a current mode controller, which is controlling a motor, as shown below.

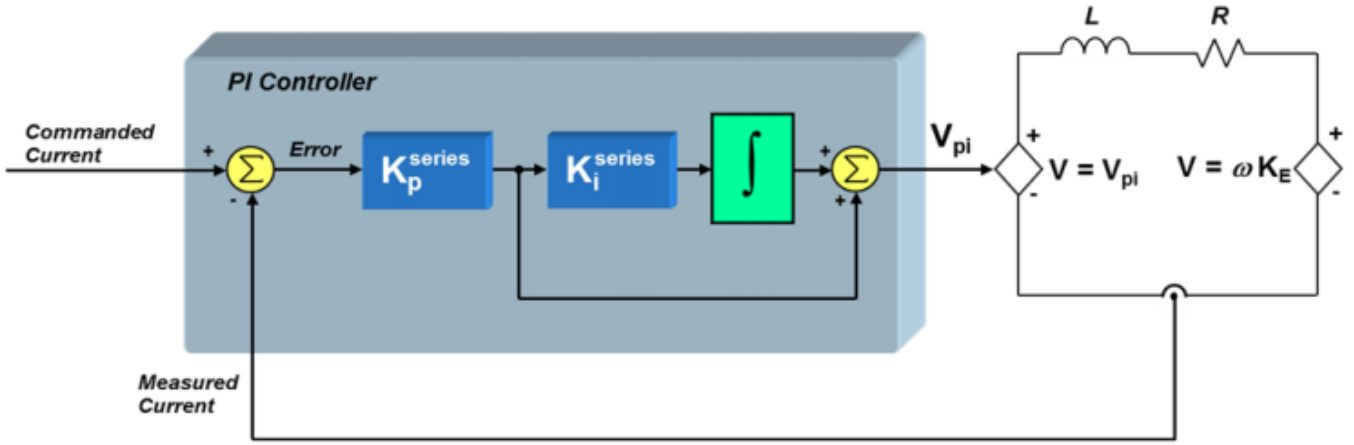



Figure 18: PI current controlled motor system including the stator.

We will use a first-order approximation of the motor winding to be a simple series circuit containing a resistor, an inductor, and a back-EMF voltage source. Assuming that the back-EMF voltage is a constar for now (since it usually changes slowly with respect to the current), we can define the small-signal transfer function from motor voltage to motor current as:

$$\frac{I(s)}{V(s)} = \frac{\frac{1}{R}}{\left(1 + \frac{L}{R}s\right)}$$

Equation 2

이제 실제 모터가 연결되서 구동된다고할 때
이 부분을 이해하기 위해서 다음 페이지의 수식들을 기술하기 시작했다.

If we also assume that the bus voltage and PWM gain scaling are included in the $K_p^{series}$ term, we can now define the "loop gain" as the product of the PI controller transfer function and the V-to-I transfer function of the RL circuit :

$$G_{loop}(s) = PI(s) \cdot \frac{I(s)}{V(s)} = \left( \frac{K_p^{series} \cdot K_i^{series} \left(1 + \frac{s}{K_i^{series}}\right)}{s} \right) \left( \frac{\frac{1}{R}}{\left(1 + \frac{L}{R}s\right)} \right)$$

Equation 3

To find the total system response (closed-loop gain), we must use the equation below:

$$G(s) = \frac{G_{loop}(s)}{1 + G_{loop}(s)}$$

(assuming the feedback term H(s) = 1)

Equation 4

첫 줄의 시도는 AC 해석을 해볼려다가 안될것 같아 바로 접고
회로를 미분 방정식으로 기술한 이후 라플라스 변환을 적용해서 수식을 유도해보고자 하였떠니 잘 진행되었다.

$$Z_{eq} = R + j\omega L \qquad i_s Z_{eq} = V \Leftrightarrow i_s = \frac{V}{Z_{eq}} = \frac{V}{R + j\omega L}$$

$$V_{pi} = L\frac{di_L}{dt} + i_L R \Leftrightarrow V_{pi}(s) = L\{sI(s) - i(0)\} + RI(s) = V(s) = LsI(s) + RI(s) = (Ls + R)I(s)$$

$$V(s) = (Ls + R)I(s), \qquad \frac{I(s)}{V(s)} = I(s)\frac{1}{(Ls + R)I(s)} = \frac{1}{(Ls + R)}$$

$$\frac{I(s)}{V(s)} = \frac{1}{(Ls + R)} = \frac{\frac{1}{R}}{\frac{L}{R}s + 1}$$

$$G_{loop} = \left\{\frac{K_p^{series} \cdot K_i^{series}\left(1 + \frac{s}{K_i^{series}}\right)}{s}\right\}\left(\frac{\frac{1}{R}}{\frac{L}{R}s + 1}\right) = \frac{\frac{K_p^{series} \cdot K_i^{series}}{R} + \frac{K_p^{series} s}{R}}{\frac{L}{R}s^2 + s} = \frac{\frac{K_p^{series}\left(s + K_i^{series}\right)}{R}}{\frac{Ls^2 + Rs}{R}} = \frac{K_p^{series}\left(s + K_i^{series}\right)}{(Ls + R)s}$$

$$G(s) = \frac{G_{loop}}{1 + G_{loop}} = \frac{\frac{K_p^{series}\left(s + K_i^{series}\right)}{(Ls + R)s}}{1 + \frac{K_p^{series}\left(s + K_i^{series}\right)}{(Ls + R)s}} = \frac{\frac{K_p^{series}\left(s + K_i^{series}\right)}{(Ls + R)s}}{\frac{K_p^{series}\left(s + K_i^{series}\right) + (Ls + R)s}{(Ls + R)s}} = \frac{K_p^{series}\left(s + K_i^{series}\right)}{K_p^{series}\left(s + K_i^{series}\right) + (Ls + R)s}$$

$$\frac{1 + \frac{s}{K_i^{series}}}{\left(\frac{L}{K_p^{series} \cdot K_i^{series}}\right)s^2 + \left(\frac{R}{K_p^{series} \cdot K_i^{series}} + \frac{1}{K_i^{series}}\right)s + 1}$$

$$G(s) = \frac{G_{loop}}{1 + G_{loop}} = \frac{\frac{K_p^{series}\left(s + K_i^{series}\right)}{(Ls + R)s}}{1 + \frac{K_p^{series}\left(s + K_i^{series}\right)}{(Ls + R)s}} = \frac{\frac{K_p^{series}\left(s + K_i^{series}\right)}{(Ls + R)s}}{\frac{K_p^{series}\left(s + K_i^{series}\right) + (Ls + R)s}{(Ls + R)s}} = \frac{K_p^{series}\left(s + K_i^{series}\right)}{K_p^{series}\left(s + K_i^{series}\right) + (Ls + R)s}$$

$$= \frac{K_p^{series} \cdot s + K_p^{series} \cdot K_i^{series}}{K_p^{series} \cdot s + K_p^{series} \cdot K_i^{series} + Ls^2 + Rs} = \frac{1 + \frac{s}{K_i^{series}}}{\frac{s}{K_i^{series}} + 1 + \frac{Ls^2}{K_p^{series} \cdot K_i^{series}} + \frac{Rs}{K_p^{series} \cdot K_i^{series}}}$$

$$= \frac{1 + \frac{s}{K_i^{series}}}{\frac{s}{K_i^{series}} + 1 + \frac{Ls^2}{K_p^{series} \cdot K_i^{series}} + \frac{Rs}{K_p^{series} \cdot K_i^{series}}} = \frac{1 + \frac{s}{K_i^{series}}}{\left(\frac{L}{K_p^{series} \cdot K_i^{series}}\right)s^2 + \left(\frac{R}{K_p^{series} \cdot K_i^{series}} + \frac{1}{K_i^{series}}\right)s + 1}$$

$$\left(\frac{L}{K_p^{series} \cdot K_i^{series}}\right)s^2 + \left(\frac{R}{K_p^{series} \cdot K_i^{series}} + \frac{1}{K_i^{series}}\right)s + 1 = (1 + Cs)(1 + Ds)$$

$$C \cdot D = \frac{L}{K_p^{series} \cdot K_i^{series}}, \qquad C + D = \frac{R}{K_p^{series} \cdot K_i^{series}} + \frac{1}{K_i^{series}}$$

$$C = \frac{R}{K_p^{series} \cdot K_i^{series}}, \qquad D = \frac{1}{K_i^{series}}$$

$$G(s) = \frac{1 + \frac{s}{K_i^{series}}}{\left(1 + \frac{R}{K_p^{series} \cdot K_i^{series}}s\right)\left(1 + \frac{s}{K_i^{series}}\right)}, \qquad \frac{R}{K_p^{series} \cdot K_i^{2 \cdot series}} = \frac{L}{K_p^{series} \cdot K_i^{series}}$$

$$\frac{R}{K_p^{series} \cdot K_i^{2 \cdot series}} = \frac{L}{K_p^{series} \cdot K_i^{series}} \Leftrightarrow \frac{R}{K_i^{series}} = L \Leftrightarrow K_i^{series} = \frac{R}{L}$$

$$G(s) = \frac{1 + \frac{s}{K_i^{series}}}{\left(\frac{L}{K_p^{series} \cdot K_i^{series}}\right)s^2 + \left(\frac{R}{K_p^{series} \cdot K_i^{series}} + \frac{1}{K_i^{series}}\right)s + 1} = \frac{1 + \frac{s}{K_i^{series}}}{\left(1 + \frac{R}{K_p^{series} \cdot K_i^{series}}s\right)\left(1 + \frac{s}{K_i^{series}}\right)}$$

$$\frac{R}{K_p^{series} \cdot K_i^{2 \cdot series}} = \frac{L}{K_p^{series} \cdot K_i^{series}} \Leftrightarrow \frac{R}{K_i^{series}} = L \Leftrightarrow K_i^{series} = \frac{R}{L}$$

$$G(s) = \frac{1 + \frac{s}{K_i^{series}}}{\left(1 + \frac{R}{K_p^{series} \cdot K_i^{series}}s\right)\left(1 + \frac{s}{K_i^{series}}\right)} = \frac{1 + \frac{Ls}{R}}{\left(1 + \frac{Rs}{K_p^{series}}\frac{L}{R}\right)\left(1 + \frac{Ls}{R}\right)} = \frac{1}{1 + \frac{L}{K_p^{series}}s}$$

$$G(s) = \frac{1}{1 + \frac{L}{K_p^{series}}s} = \frac{1}{\frac{K_p^{series} + Ls}{K_p^{series}}} = \frac{K_p^{series}}{K_p^{series} + Ls}$$

$$C \cdot D = \frac{L}{K_p^{series} \cdot K_i^{series}} = \frac{L}{K_p^{series}}\frac{L}{R}, \qquad C + D = \frac{R}{K_p^{series} \cdot K_i^{series}} + \frac{1}{K_i^{series}} = \frac{L}{K_p^{series}} + \frac{L}{R}$$

$$C = \frac{R}{K_p^{series} \cdot K_i^{series}} = \frac{L}{K_p^{series}}, \qquad D = \frac{1}{K_i^{series}} = \frac{L}{R}$$

$$\frac{R}{K_p^{series} \cdot K_i^{2 \cdot series}} = \frac{L}{K_p^{series} \cdot K_i^{series}} \Leftrightarrow \frac{R}{K_i^{series}} = L \Leftrightarrow K_i^{series} = \frac{R}{L}$$

$$L\frac{di}{dt} \Rightarrow LsI(s), \qquad iR \Rightarrow RI(s), \qquad iR + L\frac{di}{dt} \Rightarrow LsI(s) + RI(s) \Rightarrow \frac{RI(s)}{(Ls + R)I(s)} = \frac{R}{Ls + R} = \frac{1}{1 + \frac{L}{R}s} = G(s)$$

$$G(s) = \frac{1}{\frac{L}{K_p^{series}}s + 1} \Rightarrow K_p^{series} = L \cdot Bandwidth$$

**이후 계산한 수치에 입각하여 값을 설정해보니 상당히 계산 결과와 근사한 결과를 보이며 잘 동작함을 확인할 수 있었다.**