碩士學位論文

모터 전용 라이브러리를 이용한 전기자동차용 BLDC 모터 제어에 관한 연구

A Study on BLDC Motor Control for Electric Vehicle using Motor Library

國民大學校 大學院

電子工學科

盧 宣 希

2013

碩士學位論文

모터 전용 라이브러리를 이용한 전기자동차용 BLDC 모터 제어에 관한 연구

A Study on BLDC Motor Control for Electric Vehicle using Motor Library

指導敎授 安 鉉 植

이 論文을 碩士學位 請求論文으로 提出함

2013 年 12 月

國民大學校 大學院
電子工學科
盧 宣 希

2013

盧 宣 希의
碩士學位 請求論文을 認准함

2013 年 12 月

審査 委員長 金 基 斗 ㊞

審査 委員 安 鉉 植 ㊞

審査 委員 鄭 求 珉 ㊞

國民大學校 大學院

목 차

그림 목록

표 목록

국문요약

본 논문에서는 AUTOSAR 플랫폼에서 기존 안정화된 장치들과 호환성을 가지고, 직접 사용 가능한 Complex Device Driver(CDD) 로서 개발된 모터 전용 라이브러리를 사용하여 BLDC 모터 제어시스템을 구현한다. 또한 BLDC 모터 구동을 위해 CDD 와 MCAL 을 통합한 BC 알고리즘을 구현하고, BLDC 모터 구동 특성과 실제 실험을 비교함으로써 성능을 검토한다.

BLDC 모터는 전기자동차 내 서스펜션, 도어 개폐 및 연료펌프, 구동모터 등의 용도로 사용된다. 현재 하이브리드 전기자동차(HEV)를 중심으로 구동용 BLDC 모터가 적용되고 있고, 향후 기존 엔진을 완전 대체한 순수전기자동차(BEV)로 발전하기 위해 구동용 모터로 BLDC 모터를 주목하고 있다. 현재 AUTOSAR 기반으로 개발된 소프트웨어 플랫폼은 모터 구동을 위한 설계 방안, 모터 제어 등을 위해 개발되고 있다. 따라서 AUTOSAR 기반의 모터 응용 어플리케이션 개발은 모터 개발의 용이성, 재사용성, 신뢰성을 확보할 수 있다.

차량용 소프트웨어 플랫폼인 AUTOSAR 는 표준화 플랫폼 및 컴포넌트 기반 기법을 활용한 차량용 소프트웨어 개발 방법론이다. AUTOSAR 소프트웨어는 각자 다른 역할을 수행하는 여러 개의 계층으로 구분된다. 이 중 CDD 계층은 AUTOSAR 에 표준화되지 않은 소프트웨어 독립체이지만, BSW 의 API 나 AUTOSAR 의 인터페이스를 통해 접근 가능하다. CDD 는 특정 인터럽트, MCU 의 주변기기, 외부 장치 등을 사용하여 MCU 에 직접 접근하고, 일반적으로 기존에 안정적으로 동작되던 장치들을 최소한의 수정으로 구현할 수 있기 때문에 CDD 계층을 통해 모터 제어시스템을 구현하는 것이 바람직하다.

AUTOSAR 표준에 부합하고 모터 전용 라이브러리와 CDD 로서 개발된 Infineon 사의 eMotor Driver 는 MCAL 계층과 통합할 수 있고, 3 상 모터 구동을 위한 전류제어기 및 서브 모듈(EmoIcu, EmoPWM, FOC Library, PA)이 소프트웨어로 구성되어 있다. 이러한 소프트웨어 모듈을 사용하여 BLDC 모터 구동에 필요한 BC 알고리즘을 구현한다.

본 논문의 AUTOSAR 의 CDD 로서 개발된 모터 전용 라이브러리인 eMotor Driver 를 이용하여 전기자동차에서 주로 사용하고 있는 BLDC 모터 제어 시스템을

구현한다. 실험 시스템 구성은 Infineon 사의 32bit MCU 인 TC1798, Syncwork 사의 인버터 보드 및 BLDC 모터로 이뤄진다. BLDC 모터는 전기적 정류 방법인 BC 알고리즘을 사용하며, BC 알고리즘은 eMotor Driver 의 Hierarchy 에 정의된 소프트웨어 모듈과 MCAL 계층 모듈을 사용하여 구현한다. 실험 결과는 BLDC 모터의 구동 특성에서 얻을 수 있는 Hall Pattern, PWM Pattern, 상(Phase) 전압을 파악하고, PWM 의 duty cycle 을 바꿀 때 모터 속도 변화에 대해 확인한다.

1. 서론

최근 자동차 내에 전장 부품의 비율이 증가하면서 ECU(Electronic Control Unit)의 개수 또한 꾸준히 증가하고 있는 추세이다. 일본 TOYOTA 렉서스 같은 고급차의 경우 약 100 개의 ECU 가 들어가고, 현대 Genesis 는 약 60 개 정도의 ECU 가 들어간다고 한다. 이처럼 ECU 의 비중이 증가하면서 이를 제어하기 위한 소프트웨어의 LOC(Lines Of Code)가 증가하면서 복잡도가 증가하고, 복잡도가 증가할수록 품질이 저하될 가능성이 높다는 것이 문제점으로 대두되고 있다. 실제로 소프트웨어의 복잡도 증가로 인한 자동차 결함 사례가 속출하고 있다. 대표적인 예로 TOYOTA 의 Prius 하이브리드 자동차는 소프트웨어의 사소한 오류 때문에 가솔린 엔진이 정지하는 문제를 가지고 있었으며 이를 해결하기 위해 총 7 만 5 천대를 리콜하여 새로운 소프트웨어를 설치하여야 했다. 또한, BMW 사의 745i 도 엔진 밸브의 타이밍을 조정하는 두 ECU 사이의 동기화 결함으로 주행 중 엔진이 꺼지는 문제가 있어 약 5 천만대가 리콜된 바 있다. 이러한 복잡성 문제를 해결하기 위해 자동차용 소프트웨어의 표준화 필요성이 증가하고 있다[1,2].

AUTOSAR(AUTomotive Open System ARchitecture)는 표준화 플랫폼 및 컨포넌트 기반 기법을 활용한 차량용 소프트웨어 개발 방법론이며, AUTOSAR 컨소시엄에 속한 완성차 업체 및 부품업체들의 AUTOSAR 표준사양의 양산적용 사례가 늘어가고 있는 추세이다[3]. AUTOSAR 소프트웨어는 크게 AUTOSAR SWC(Software Component), RTE(Run Time Environment), BSW(Basic Software), MCAL(Microcontroller Abstraction Layer), CDD(Complex Device Drivers) 계층으로 구분된다. 여기서 MCAL 은 MCU(Micro-Controller Unit)의 실제 물리적 신호에 대한 접근을 제공하며, 하드웨어 의존성이 있는 부분으로 통신 드라이버, ADC(Analog to Digital Convert), PWM(Pulse Width Modulation), DIO(Digital Input Output) 등의 I/O 드라이버, EEPROM(Electrically Erasable and Programmable Read Only Memory), Flash 등의 메모리 드라이버 등으로 구성된다[4]. AUTOSAR 의 CDD 계층은 AUTOSAR 에 표준화되지 않은 소프트웨어 독립체이지만, BSW 의 API 나 AUTOSAR 의 인터페이스를 통해 접근 가능하다.

CDD 의 주된 사용 목적은 특정 인터럽트, MCU 의 주변기기, 외부 장치 등을 사용하여 MCU 에 직접 접근하고, 일반적으로 기본 하드웨어를 직접 제어해야 하는 복잡한 센서 또는 신뢰성을 중요시 하는 파워트레인의 연료 주사, 전자벨브 제어, 미션 제어 등의 영역에서 사용하기 위함이다[5]. CDD 계층을 사용하면 하드웨어에 직접 접근하여 제어 시스템을 구동시킬 수 있기 때문에 시스템 실행 시간, 데이터 전달 시간이 향상되고 메모리 사용이 줄어들게 된다[6].

AUTOSAR 기반으로 개발된 소프트웨어 플랫폼은 Freescale 사의 AUTOSARFS, FUJITSU 사의 AUTOSAR ROADSTER, TI 사의 Hercules 등이 있다. 한 예로 Freescale 사의 AUTOSARFS 는 AUTOSAR 호환 OS(Operating System)와 MCAL 구성요소를 모두 제공한다. BSW 와 MCAL 을 사용하여 모터 제어시스템 구현이 가능하지만, 모터 제어에 필요한 제어기, 벡터제어 알고리즘 등은 핸드코딩이 필요하다. 모터 제어시스템은 BSW 를 통해서 구현할 수 있지만 CDD 를 통해서 구현할 수 있다. CDD 계층은 기존에 안정적으로 동작되던 장치들을 최소한의 수정으로 구현할 수 있기 때문에 모터 구동에 필요한 드라이버를 추가하여 사용할 수 있다. 따라서 모터 전용으로 개발된 CDD 를 통해 모터 제어시스템을 구현하는 것은 핸드코딩을 최소화하고 BSW 를 사용하는 것 보다 시스템 실행시간을 절약할 수 있다.

AUTOSAR 표준에 부합하고 모터 전용 라이브러리로서 개발된 Infineon 사의 eMotor Driver 는 AUTOSAR CDD 에 해당하고, 3 상 모터 구동을 위한 벡터제어 기법, 전류제어기 및 서브 모듈 등이 소프트웨어로 구성되어 있다. MCU 에 종속적이지만 직접접근이 가능한 CDD 를 이용하여 BSW 에 접근할 수 있기 때문에 BSW 서비스 일부를 사용할 수 있다. 또한 Infineon 사의 32bit MCU 인 TriCore 에 대하여 안전성 플랫폼을 이용할 수 있어 ISO26262 에 따른 ASIL(Automotive Safety Integrity Level) B-D 요구에 관한 하드웨어 및 소프트웨어 측면을 충족한다. 따라서 eMotor Driver 를 이용한 모터 제어 시스템 구현 시 AUTOSAR 표준에 따른 각 소프트웨어의 독립적 개발로 인하여 제어기 설계와 재 설계가 용이하다[7].

BLDC(Brushless DC) 모터는 일반적인 DC 모터에 비해 관성 대 토크 비율이 높고, 브러시가 없음으로 브러시와 정류자의 마찰에 의해 발생하는 스파크 및

마모 그리고 가청잡음이 없는 장점을 가지고 있다. BLDC 모터는 전기자동차 내 서스펜션, 도어 개폐 및 연료펌프, 구동모터 등의 용도로 사용된다. 현재 하이브리드 전기자동차(HEV)를 중심으로 구동용 BLDC 모터가 적용되고 있고, 향후 기존 엔진을 완전 대체한 순수전기자동차(BEV)로 발전하기 위해 구동용 모터로 BLDC 모터를 주목하고 있다[8]. 현재 AUTOSAR 기반으로 개발된 소프트웨어 플랫폼은 모터 구동을 위한 설계 방안, 모터 제어 등을 위해 개발되고 있다. 따라서 AUTOSAR 기반의 모터 응용 어플리케이션 개발은 모터 개발의 용이성, 재사용성, 신뢰성을 확보할 수 있다.

본 논문에서는 AUTOSAR 의 CDD 로서 개발된 모터 전용 라이브러리인 eMotor Driver 를 이용하여 전기자동차에서 주로 사용하고 있는 BLDC 모터 제어 시스템을 구현한다. 실험 시스템 구성은 32bit MCU, 인버터 보드 및 BLDC 모터로 이뤄진다. BLDC 모터는 전기적 정류 방법인 BC 알고리즘을 사용하며, BC 알고리즘은 eMotor Driver 의 Hierarchy 에 정의된 소프트웨어 모듈과 MCAL 계층 모듈을 통합하여 구현한다. 실험 결과는 BLDC 모터의 구동 특성에서 얻을 수 있는 Hall Pattern, PWM Pattern, 상(Phase) 전압을 파악하여 실제 실험과 비교하고, PWM 의 duty cycle 을 바꿀 때 모터 속도 변화에 대해 확인한다.

본 논문은 모두 5 장으로 구성된다. 제 2 장에서는 BLDC 모터의 동작 특성과 제어 방법에 대하여 설명하고, 제 3 장에서는 모터 전용 라이브러리 개요와 이를 이용한 제어시스템의 성능 분석에 대하여 설명한다. 제 4 장에서는 실험 환경 구축 및 실험 결과를 보인다. 그리고 5 장에서는 본 논문의 결론을 기술한다.

2. BLDC 모터 제어시스템 구성 및 제어방법

기존의 DC 모터는 기계적인 접촉 구조인 정류자와 브러시를 이용하여 회전에 따라 전기자 전류의 극성을 바꾸어 주어야만 일정한 방향으로 토크가 발생하게 되어 연속적인 회전이 가능하다. 이러한 구조적 특성은 브러시 수명, 브러시 잔류물, 최대 속도 및 전기 노이즈 그리고 기계적 소음 등의 원인으로 여러 한계를 갖는다. BLDC(Brushless DC) 모터는 회전자에 권선을 감는 대신 영구자석을 장착하여 일반적인 DC 모터에 비해 관성 대 토크 비율이 높고, DC 모터에서 브러시를 제거하여 브러시 로드의 마찰, 스파크 및 마모 그리고 가청잡음을 피함으로써 기존 DC 모터보다 빠르고 효율적인 동작 성능을 제공한다. BLDC 모터는 DC 모터에 비하여 소형화, 경량화가 이루어졌고, 각 응용에 적합하게 센서 일체형으로 저장되어 있으므로 디스크 드라이브와 고효율, 유연한 동작 그리고 정밀한 속도 제어가 필요한 산업용 이동 장비, 차량 내 전장부품 등에 사용된다[8].

본 논문에서는 AUTOSAR 의 CDD 로서 개발된 모터 전용 라이브러리를 이용하여 전기자동차에서 주로 사용하고 있는 BLDC 모터 제어 시스템을 구현한다. BLDC 모터는 전기자동차 내 서스펜션, 도어 개폐 및 연료펌프, 구동모터 등의 용도로 사용된다. BLDC 모터는 일반적인 DC 모터에 비해 관성 대 토크 비율이 높고, 브러시가 없음으로 브러시와 정류자의 마찰에 의해 발생하는 스파크 및 마모 그리고 가청잡음이 없는 장점을 가지고 있다[8]. 본 장에서는 BLDC 모터 구동을 위한 BLDC 모터 동작 특성과 구동 원리에 대해 살펴본다.

2.1 BLDC 모터 동작 특성

DC 모터는 기계적인 접촉 구조인 정류자와 브러시를 이용하여 회전에 따라 전기자 전류의 극성을 바꾸어 주어야만 일정한 방향으로 토크가 발생하게 되어 연속적인 회전이 가능하다. 이러한 정류자와 브러시는 회전 시 전자기적 잡음과 기계적 소음이 발생하며, 마찰에 따른 마모로 인해 정기적인 유지보수가 필요하다. 그러나 BLDC 모터는 회전자의 위치 정보를 사용하여 DC 모터의

정류자와 브러시 기능을 반도체 스위치 소자로 구현함으로써 DC 모터의 단점을 해결한 모터이다. 이와 같은 BLDC 모터는 많은 도체로 이루어진 무거운 전기자가 회전하는 DC 모터에 비해 영구자석이 회전하는 구조이므로 관성이 작게 되어 빠른 가감속에 유리하다. 또한 권선이 고정자 측에 있어 방열이 용이하므로 온도에 따른 영구자석의 감자를 피하기 위해 최대 전기자 전류에 제약이 있는 DC 모터보다 최대 출력 토크 발생 측면에서 유리하다. 또한 BLDC 모터는 DC 모터에서와 같은 정류자와 브러시의 기계적인 접촉문제와 정류기능의 문제가 없으므로 고속 운전이 가능하다는 장점을 갖고 있다[8,9].

3 상 2 극 Y 결선 BLDC 모터의 구동 시스템을 그림 2.1 에 나타내었다. BLDC 모터에서는 정류작용을 위한 브러시 및 정류자가 필요 없는 대신에 회전자의 위치를 검출하는 센서와 이 위치 정보에 따라 해당하는 고정자 코일에 스위칭을 통해 전류를 흘려주는 소자가 필요하다. 그림 2.2 에서는 위치 검출 소자로 홀(Hall) 소자를 사용하고 스위칭 소자로 트랜지스터를 사용한 예이다. 그림 2.1 에서와 같이 3 개의 홀 소자는 전기적으로 120° 의 위상차를 가지고 있어 6 개 구간으로 구별하여 검출할 수 있다. 검출된 회전자의 위치 정보를 사용하여 연속적인 회전을 위해 필요한 두 상의 전류를 인버터를 통해 고정자 권선에 흘려준다[9].

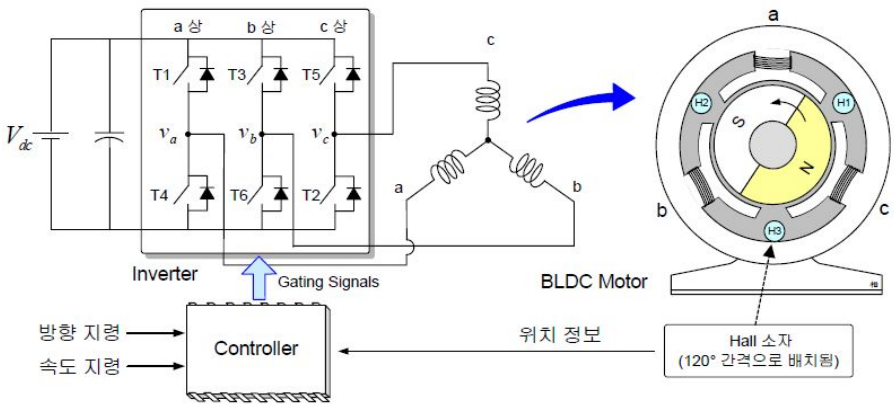


그림 2.1 BLDC 모터의 구동 시스템.

Fig. 2.1 The Driving System of BLDC Motor.

2.2 BLDC 모터의 구동 원리

BLDC 모터의 제어 방법에는 여러 가지가 있지만 본 논문에서 사용된 방법은 전자적인 정류(Block Commutation)이다. 본 절에서는 전자적인 정류에 대해 설명하고, 이에 필요한 Hall Pattern 과 PWM Pattern 에 대해 파악한다. BLDC 모터를 위한 인버터는 통상 어느 시점에 두 개의 스위칭 소자만이 동작하는 2 상 여자 방식으로 구동된다. 각 스위칭 소자는 120° 씩 통전하며, 암(Arm) 단락 방지를 위한 데드 타임이 필요 없다. 3 상 2 극 BLDC 모터의 구동 과정이 그림 2.2, 2.3, 2.4 에 보인다.

홀 소자(H1, H2, H3)는 N 극 검출 시에는 1 을, S 극 검출 시에는 0 을 출력한다고 가정한다. 그림 2.2 의 1 을 보면 6 개의 스위치 중 T1 과 T6 가 도통되어 a 상과 b 상을 통해 전류가 흐른다. 권선에 흐르는 전류에 의해서 a 상은 S 극, b 상은 N 극을 띄게 되면, 정지하고 있던 영구자석이 반 시계 방향으로 60° 회전하게 된다. 각각 홀 소자는 H1 에는 1, H2 에는 0, H3 에는 1 을 검출하고, 101_2(H3, H2, H1)의 이진수를 십진수로 변환하면 5 가 된다.

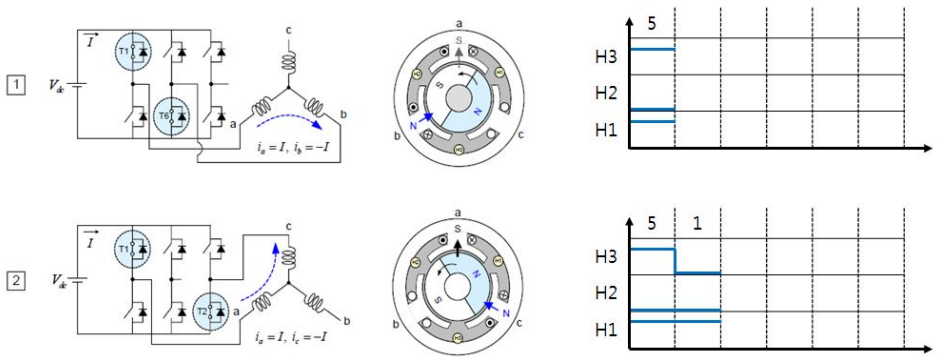


그림 2.2 3 상 2 극 BLDC 모터의 구동 과정(1/3).

Fig. 2.2 The Driving Process of three-phase and two-pole BLDC Motor(1/3).

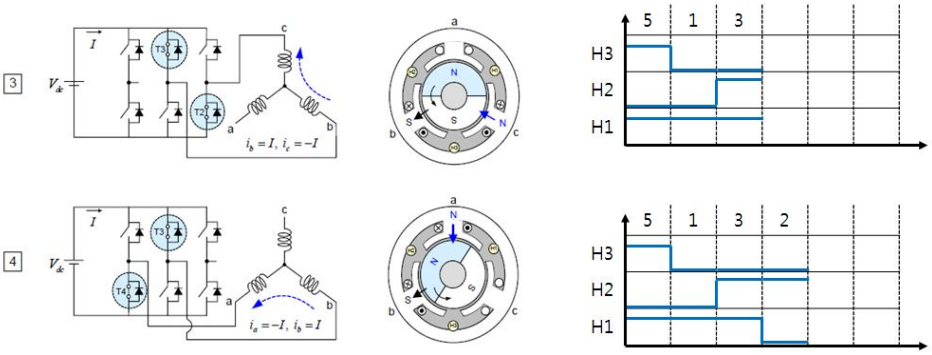


그림 2.3 3상 2극 BLDC 모터의 구동 과정(2/3).

Fig. 2.3 The Driving Process of three-phase and two-pole BLDC Motor(2/3).

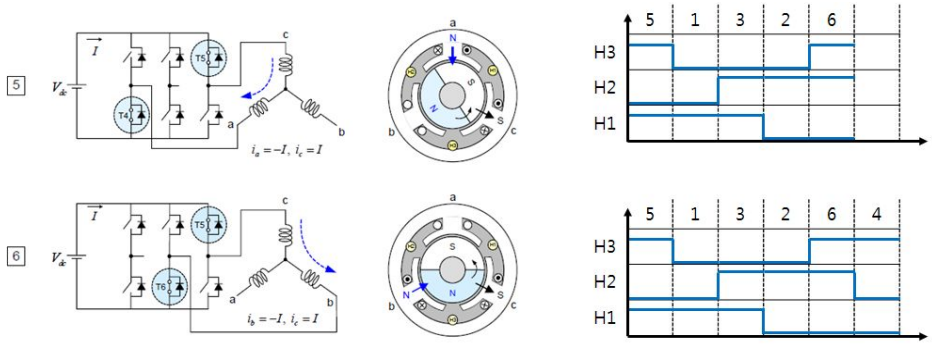


그림 2.4 3상 2극 BLDC 모터의 구동 과정(3/3).

Fig. 2.4 The Driving Process of three-phase and two-pole BLDC Motor(3/3).

이와 마찬가지로 그림 2.3, 2.4를 살펴보면 반시계 방향으로 회전하기 위해 필요한 Hall Pattern(5→1→3→2→6→4)과 PWM Pattern을 알 수 있고 이를 정리한 것은 표 2.1과 같다[10].

표 2.1 BLDC 모터의 Hall Pattern 과 PWM Pattern

Table 2.1 The Hall Pattern and PWM Pattern of BLDC Motor

구분		1	2	3	4	5	6
Hall Pattern	Hall Value (H3/ H2/ H1)	5	1	3	2	6	4
	Next Required Hall Value	1	3	2	6	4	5
PWM Pattern	T2	0	1	1	0	0	0
	T5	0	0	0	0	1	1
	T6	1	0	0	0	0	1
	T3	0	0	1	1	0	0
	T4	0	0	0	1	1	0
	T1	1	1	0	0	0	0

BLDC 모터의 전압 방정식은 식(1)과 같고, 고정자 전압 v_{abc}, 상 전류 i_{abc}, 고정자 저항 R_s, 역기전력 (Trapezoidal Back EMF) e_{abc}, 고정자 인덕턴스 L_s, 상호 인덕턴스 L_m로 이루어져 있다. BLDC 모터의 출력 전력 P_e 은 식(2)와 같고, 회전자 각 속도 ω_m 로 부터 일반화된 BLDC 모터의 토크는 식(3)과 같다.

$$v_{abc} = R_s i_{abc} + (L_s - L_m)\frac{di_{abc}}{dt} + e_{abc} \quad (1)$$

$$P_e = e_a i_a + e_b i_b + e_c i_c \quad (2)$$

$$T_e = \frac{P_e}{\omega_m} = \frac{e_a i_a + e_b i_b + e_c i_c}{\omega_m} \quad (3)$$

BLDC 모터가 그림 2.4 와 같이 반시계 방향으로 회전함에 따라 발생되는 Hall Pattern, 역기전력, 상 전류 및 토크는 그림 2.5 에서 확인할 수 있다. 또한 BLDC 모터의 영구자석은 고정자 권선에 흐르는 구형파 전류와 작용하여 일정한

토크를 발생시키기 위해 그림 2.5 와 같이 근사 사다리꼴의 역기전력를 갖는 것을 확인할 수 있다[9].

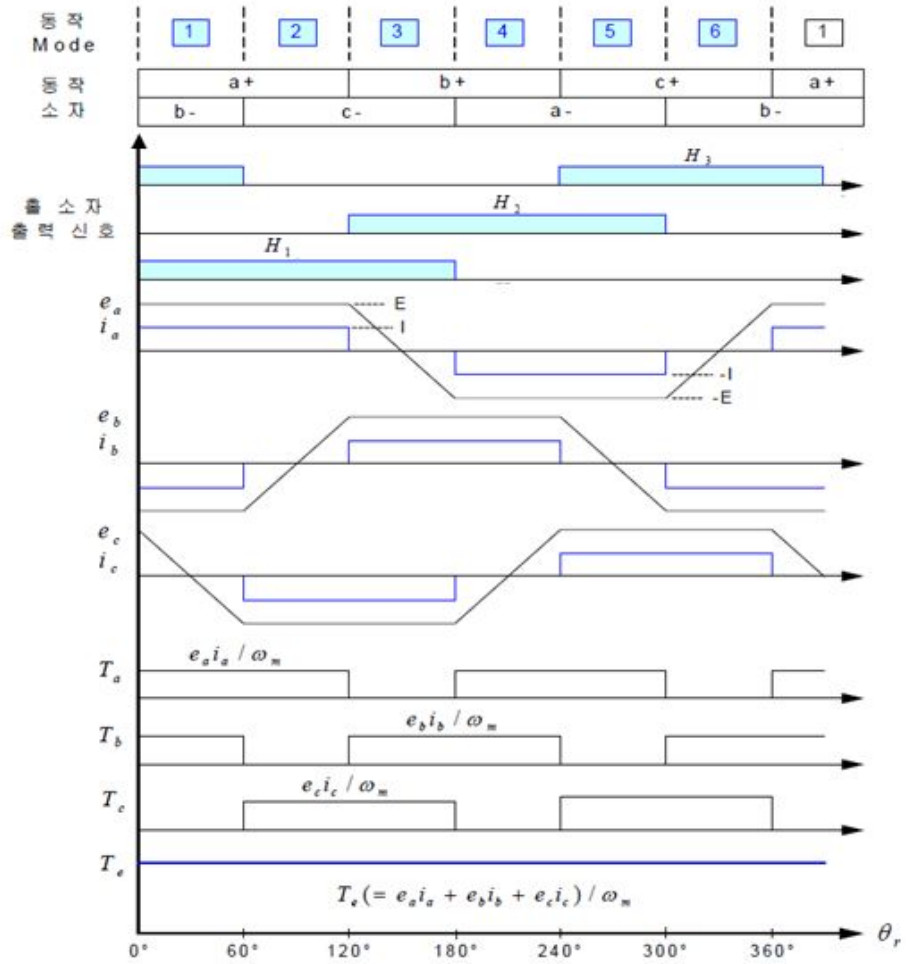


그림 2.5 3상 2극 BLDC 모터의 구동 원리.

Fig. 2.5 The Driving Principle of three-phase and two-pole BLDC motor.

3. 모터 전용 라이브러리 기반 제어시스템의 성능 분석

일반적인 모터 제어알고리즘은 C 언어 또는 C++을 사용하여 개발자가 직접 전류제어기, 속도검출 알고리즘 등을 개발하기 때문에 해당 제어 시스템의 소프트웨어 알고리즘은 알고리즘의 안전성, 호환성 그리고 신뢰성 검증이 어렵다는 단점이 있었다. 또한 기존 제어 시스템 설계 방법은 설계 프로세스의 디버깅이나 마지막 단계에서 결과를 확인해야 했기 때문에 문제점을 확인 및 수정하기까지의 시간과 비용이 많이 소모되었다. 이러한 설계 방법에서의 설계 변경은 시스템의 일부분 또는 모든 부분을 반복적으로 재검토하고, 다른 에러를 유발 시키는 등의 문제를 가지고 있다. 그러나 모터 전용 라이브러리를 이용한 모터 제어시스템 개발 방법은 검증된 프로그램과 표준화된 플랫폼을 사용함으로써 차량용 소프트웨어의 안전성, 호환성, 신뢰성을 보장한다.

모터 전용 라이브러리는 모터 제어를 위한 다양한 알고리즘과 어플리케이션을 제공하여 효율적인 모터 제어가 가능하다. 또한 최적화되고 신뢰성 있는 어플리케이션을 사용함으로써 개발 시간이 단축되고, 복잡도를 낮춘다. 본 논문에서는 모터 전용 라이브러리와 AUTOSAR 의 CDD 로서 개발된 eMotor Driver 를 사용한다. 따라서 본 장에서는 모터 전용 라이브러리를 차량용 소프트웨어 플랫폼인 AUTOSAR (Automotive Open System Architecture) 측면에서 해석하고, 모터 제어시스템 적용 방법에 대하여 설명한다.

3.1 AUTOSAR

급격하게 증가하는 차량 내 내장형 제어시스템들의 규모는 점차 증가하고 있으며 소프트웨어 복잡도 역시 기하급수적으로 증가하고 있다. 이러한 복잡한 소프트웨어를 빠르고 신뢰성 있게 개발하기 위하여 AUTOSAR 는 차량용 소프트웨어 아키텍처와 개발방법론 등의 표준 명세를 정의하였다. AUTOSAR 표준에서 정의하는 소프트웨어 구조는 그림 3.1 과 같이 계층 구조를 기초로 하고 있으며 크게 SWC(Software Component), RTE(Run Time Environment),

BSW(Basic Software), MCAL(Micro-Controller Abstraction Layer), CDD(Complex Device Driver) 등으로 구성되어 있다.

각각의 계층들은 표준화 된 API(Application Program Interface)를 이용하여 연결이 가능하며, 각 계층은 다른 계층에 독립적으로 구현 될 수 있기 때문에 확장성이 확보된다. 또한 AUTOSAR 는 복잡한 차량용 소프트웨어를 아키텍처와 개발방법론 등의 표준 명세를 정의하여 신뢰성을 높이며, 런타임 환경 하에 하드웨어와 소프트웨어 구성요소를 RTE 라는 미들웨어로 분리시킴으로써 하드웨어에 대한 의존성을 최소화하고, 소프트웨어의 재사용성을 향상 시킨다[11].

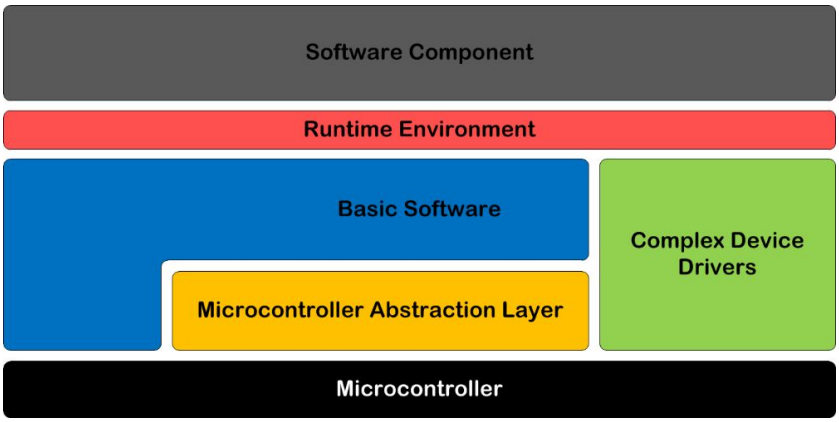


그림 3.1 AUTOSAR 의 계층 구조.

Fig. 3.1 The Layers Structure of AUTOSAR Software.

SWC 는 하드웨어와 네트워크에 의존적인 정보들은 배제하고 어플리케이션의 기능적인 요소를 포함하는 계층이다. SWC 는 가상의 통신 계층인 VFB(Virtual Functional Bus)를 통해 연결되어, 가상의 네트워크를 통해 서로 통신하도록 구성된다[11]. SWC 는 그림 3.2 와 같이 VFB 를 통해 연결되어 있기 때문에 하드웨어나 네트워크의 설계 변경으로 인한 의존성을 최소화하고 어플리케이션의 기능적 요소에 집중하여 개발이 이루어진다. SWC 에 ECU 배치가 결정되면 그림 3.2 과 같이 가상 계층이었던 VFB 가 각각의 ECU 별로 RTE 라는 이름의 통합 계층으로 구현된다.

RTE 는 SWC 와 BSW 를 연결하는 미들웨어이며, 포트라는 객체를 통해 컴포넌트들 사이의 연결 상태를 관리한다. 한 컴포넌트가 포트를 통해 서비스를 요청하거나 데이터를 쓰면, 그 포트에 연결된 상대편 포트는 자신이 속한 컴포넌트에게 서비스가 요청되었음을 알리거나 전송된 데이터를 제공한다. 통신할 컴포넌트에 대한 정보가 포트 객체에 숨겨져 있기 때문에, 각 컴포넌트는 상대편 컴포넌트의 위치나 존재 여부를 알 필요가 없다[4]. 또한 RTE 는 독립된 시스템을 통합하여 API 를 통해 mapping 하고 communication path 를 제공한다. VFB 와 RTE 는 공통으로 API 를 공유하지만, 기본적으로 다른 개념을 가진다. VFB 는 집중형 미들웨어 계층이며 RTE 는 분산형 미들웨어 계층이다[12].

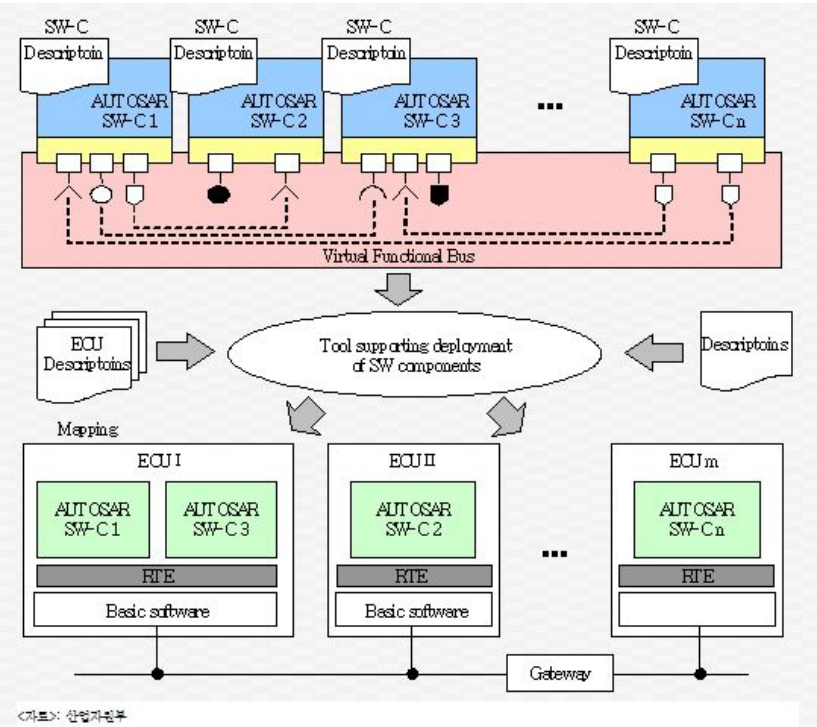


그림 3.2 VFB 를 중심으로 한 요소들의 상호 구조.

Fig. 3.2 The Interconnection structure of components focus on VFB.

BSW 는 AUTOSAR 의 최하위 계층으로 OS(Operating System), 디바이스 드라이버, 커뮤니케이션 같은 SWC 가 필요한 작업을 수행하기 위한 서비스를 제공한다. BSW 는 크게 Service Layer, ECU Abstraction Layer, MCAL, CDD 로 구분된다. Service Layer 는 BSW 의 최상위 계층으로 표 3.1 과 같은 서비스를 제공한다.

표 3.1 BSW 내 Service Layer 의 특징

Table 3.1 The Features of Service Layer in BSW

● Operating System service
● Automobile metwork communications and management services
● Memory service
● Diagnosis service
● ECU status control

ECU Abstraction Layer 는 MCAL 계층과 연결되어있고, MCU 안에 장치들이 어떻게 구성되어 있는지(메모리 맵, I/O 맵, SPI 등)에 상관없이 MCU 에 접근할 수 있도록 해준다. 반면 MCAL 계층은 BSW 의 최하위 계층으로 실제로 특정 장치(RAM, Flash, 네트워크 등)를 작동시키기 위한 장치 드라이버들을 구현하고 있는 계층이다. MCAL 계층은 MCU 의 주변 기기 메모리, 레지스터에 직접 접근이 가능하다[13].

CDD 는 다른 계층들과 다르게 시스템 타이밍과 관련된 태스크(Task)를 다루는데 사용되고, MCU 의 내부 인터럽트와 연관되어 있다. CDD 는 하드웨어에 직접 접근이 가능하며, AUTOSAR BSW 의 전 계층을 사용할 수 있다[14]. CDD 의 주된 사용 목적은 특정 인터럽트, MCU 의 주변기기, 외부 장치 등을 사용하여 MCU 에 직접 접근하고, 일반적으로 기본 하드웨어를 직접 제어해야 하는 복잡한 센서 또는 신뢰성을 중요시 하는 파워트레인의 연료 주사, 전자벨브 제어, 미션 제어 등의 영역에서 사용하기 위함이다[5]. CDD 계층을 사용하면 하드웨어에 직접 접근하여 제어 시스템을 구동시킬 수 있기 때문에 시스템 실행 시간, 데이터 전달 시간이 향상되고 메모리 사용이 줄어들게 된다[6].

AUTOSAR 기반으로 개발된 소프트웨어 플랫폼은 Freescale 사의 AUTOSARFS, FUJITSU 사의 AUTOSAR ROADSTER, TI 사의 Hercules 등이 있다. 한 예로

Freescale 사의 AUTOSARFS 는 AUTOSAR 호환 OS 와 MCAL 구성요소를 모두 제공한다. BSW 와 MCAL 을 사용하여 모터 제어시스템 구현이 가능하지만, 모터 제어에 필요한 제어기, 벡터제어 알고리즘 등은 핸드코딩이 필요하다. 모터 제어시스템은 BSW 를 통해서 구현할 수 있지만 CDD 를 통해서 구현할 수 있다. CDD 계층은 기존에 안정적으로 동작되던 장치들을 최소한의 수정으로 구현할 수 있기 때문에 모터 구동에 필요한 드라이버를 추가하여 사용할 수 있다. 따라서 모터 전용으로 개발된 CDD 를 통해 모터 제어시스템을 구현하는 것은 핸드코딩을 최소화하고 BSW 를 사용하는 것 보다 시스템 실행시간을 절약할 수 있다.

3.2 eMotor Driver

AUTOSAR 표준에 부합하고 모터 전용 라이브러리로서 개발된 eMotor Driver 는 AUTOSAR CDD 에 해당하고, 3 상 모터 구동을 위한 벡터제어 기법, 전류제어기 및 서브 모듈 등이 소프트웨어로 구성되어 있다. MCU 에 종속적이지만 직접접근이 가능한 CDD 를 이용하여 BSW 에 접근할 수 있기 때문에 BSW 서비스 일부를 사용할 수 있다. eMotor Driver 는 AUTOSAR 의 MCAL 계층과 통합이 가능하므로 신뢰성이 보장된다. 또한 GUI(Graphical User Interface)기반의 어플리케이션 개발 툴을 통해 쉽게 32bit MCU 를 기반으로 3 상의 모터를 제어할 수 있기 때문에 자동차 ECU 개발자들은 특정 어플리케이션에 따른 전기 모터 제어에 집중할 수 있게 되고, 모터 제어 알고리즘을 재프로그램 할 필요가 없다.

eMotor Driver 는 Infineon 사의 32bit MCU 인 TriCore 에 대하여 안전성 플랫폼을 구현할 수 있고, 3 상 모터를 제어하기 위한 FOC(Field Oriented Control) Mode 와 BC(Block Commutation) Mode 를 지원한다. 각 Mode 는 전류제어를 이용한 모터 제어 알고리즘을 제공하며, 전류제어 루프를 50 마이크로 초 이내에 처리한다. 또한 모터 회전자의 위치 감지를 위한 센서 Mode 로 홀 센서, 증분형 엔코더와 리졸버 Mode 를 사용한다. 직접적으로 리졸버 Mode 를 제공하므로 별도로 외부 리졸버 IC 를 필요로 하지 않아 원가를 절감할 수 있다. 따라서 eMotor Driver 를 이용한 모터 제어 시스템 구현 시 AUTOSAR 표준에 따른 각 소프트웨어의 독립적 개발로 인하여 제어기 설계와 프로그램 재설계가 용이하다[15].

표 3.2 eMotor Driver 의 특징

Table 3.2 The Features of eMotor Driver

- Supported AUTOSAR releases and devices
  - --V2.0: AUDO NG (TC1796, TC1766)
  - --V2.1, V3.0: XC2287, AUDO Future (TC1797, TC1767), AUDO S
  - --V3.1, V3.2: XC2000, AUDO MAX
  - --V4.03: AUDO MAX

--V3.2, V4.03: AURIX™

--ISO 26262 support

- Complex driver for non-standardized modules (for TriCore™)
- AUTOSAR BSW suite via partners: Electrobit, Vector, KPIT
- Delivery packages include: source code, user manual, Tresos configuration tool
- Control PMSM motors via Field Oriented Control (FOC), including Space Vector Modulation SVM
- Control BLDC motors via Block Communication (BC)
- Mixed control of FOC / BC motors
- Integrated with AUTOSAR drivers
- Supports safety applications Sensors in FOC Mode
- Hall sensors/Incremental encoder
- Direct resolver mode (without resolver IC)
- Resolver mode (with resolver IC)
- Sensorless FOC
- Current Measurement: 3 phases, 2-phase parallel and sequential, DC link sequential Sensors in BC Mode
- Hall sensors
- Sensorless via back EMF
- Current Measurement: DC link single MC-ISAR eMotor Benefits:
- Developed for mass production, off-the-shelf implementation Limited software outlay
- Direct resolver mode (no external resolver IC), reduced system cost Compliant to ISO 26262 process and CMM level 3
- Seamless configuration under the same configuration tool for AUTOSAR MCAL driver
- Easy to use

MC-ISAR eMotor Driver 의 주요 특징은 표 3.2 와 같다. AUTOSAR 에 부합하는 eMotor Driver 의 소프트웨어 구조는 그림 3.3 과 같고 모터 제어에 관련된 CDD 계층의 내부 구성은 그림 3.4 와 같다. 그림 3.3 을 보면 MCAL 계층에서 사용되는 모듈 중 일부(Infineon MC-ISAR driver)를 eMotor Driver 에서 제공하여 CDD 계층과 MCAL 계층을 통합할 수 있다.

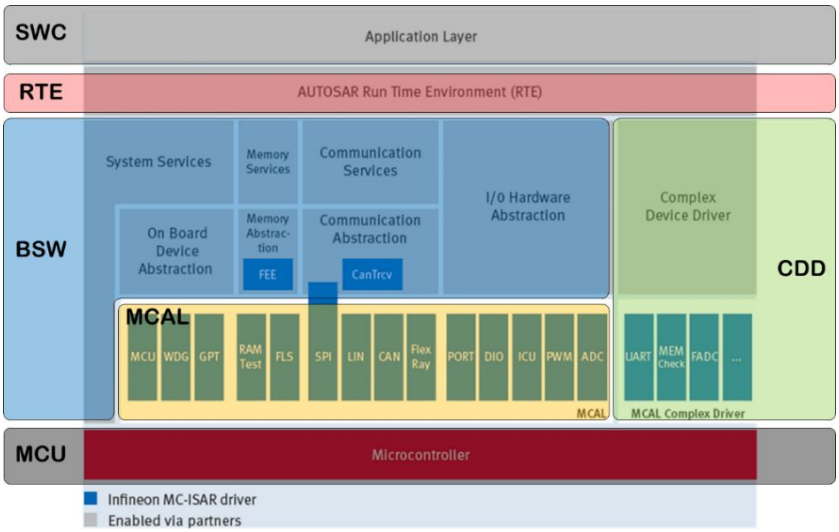


그림 3.3 MC-ISAR eMotor Driver 의 소프트웨어 구조.

Fig. 3.3 The Software Architecture of MC-ISAR eMotor Driver.

eMotor Driver 의 Hierarchy 는 그림 3.4 와 같이 3 개로 구분되며, 구현하고자 하는 모터는 FOC 혹은 BC 알고리즘 선택함에 따라 각기 다른 Hierarchy 를 사용한다. 각 Hierarchy 는 AUTOSAR 에서 제공하는 API 제어기를 적절히 사용하여 사용자가 직접 접근이 가능하다.

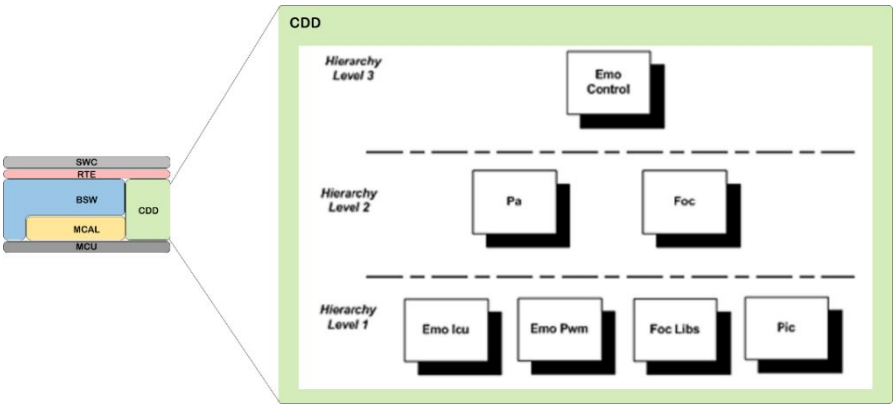


그림 3.4 eMotor Files Hierarchy.

Fig. 3.4 The eMotor Files Hierarchy.

Hierarchy Level 3(EmoControl)은 가장 상위 계층으로 FOC, BC 알고리즘을 선택하고, 모터 구동 및 정지, 모터 속도 검출 등의 API 를 사용할 수 있다.

Hierarchy Level 2(PA, FOC)는 Hierarchy Level 3 에서 선택된 알고리즘이 구현되어 있는 계층이다. PA(Position Acquisition)는 엔코더, 레졸버, 홀 센서를 이용하여 속도를 검출하는 모듈이고, FOC 는 Hierarchy Level 1 에 있는 Foc Libs(PARK, CRAKE, SWM Library)를 이용하여 FOC Mode 가 구현되어 있는 계층이다.

Hierarchy Level 1(Emo Icu, Emo Pwm, Foc Libs, PI 제어기)은 선택된 알고리즘 내에 사용되는 모듈을 활성화하고, PI 제어기 등이 구현되어 있다. 속도 검출을 위한 EmoIcu, PWM 생성을 위한 EmoPWM, 제어기 구현을 위한 PI 제어기 모듈을 사용하고, FOC Mode 내에서 사용되는 벡터제어 기법을 포함하는 Foc Libs 모듈로 구성되어 있다.

각 Hierarchy 에 존재하는 모듈에 따른 소스파일들은 그림 3.5 와 같이 정의되어있다. 해당 소스파일을 적절히 이용하여 모터 구동 및 제어가 가능하다[7].

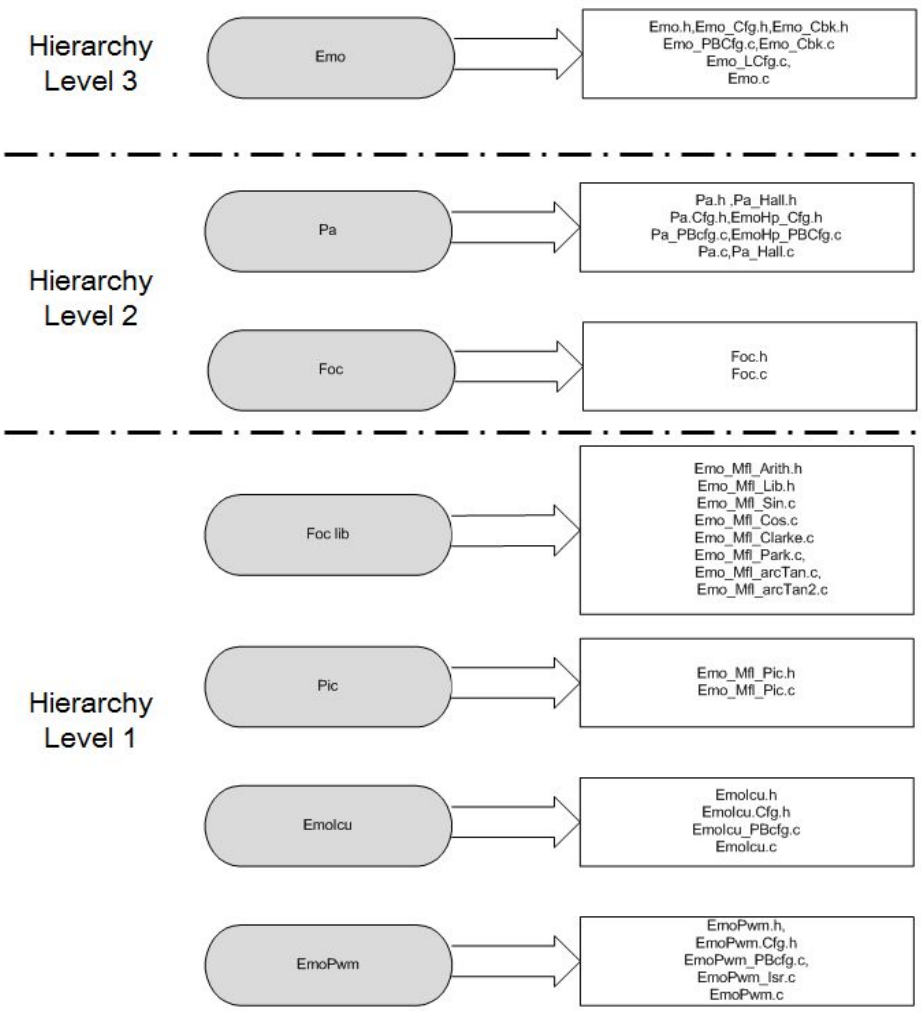


그림 3.5 eMotor Driver 의 파일 구조.

Fig. 3.5 The Structure of eMotor Files.

FOC 와 BC Mode 선택에 따라 선정되는 모듈은 그림 3.6, 3.7 과 같이 도식화 되어있다[16]. FOC Mode 는 PMSM 모터 구동을 위한 벡터제어 방법이며 그림 3.6 과 같다. [17]에는 FOC Mode 에 따른 PMSM 모터 구동 방법이 명시되어 있지만 BC Mode 를 이용한 BLDC 모터 구동 방법은 나타내지 않았다. 본 논문에서는 그림 3.7 과 같이 BC Mode 의 모듈 구성도를 기반으로 BLDC 모터 구동을 위한 Hierarchy 를 배치하고, API 를 선정하여 BC Mode 를 구현한다.

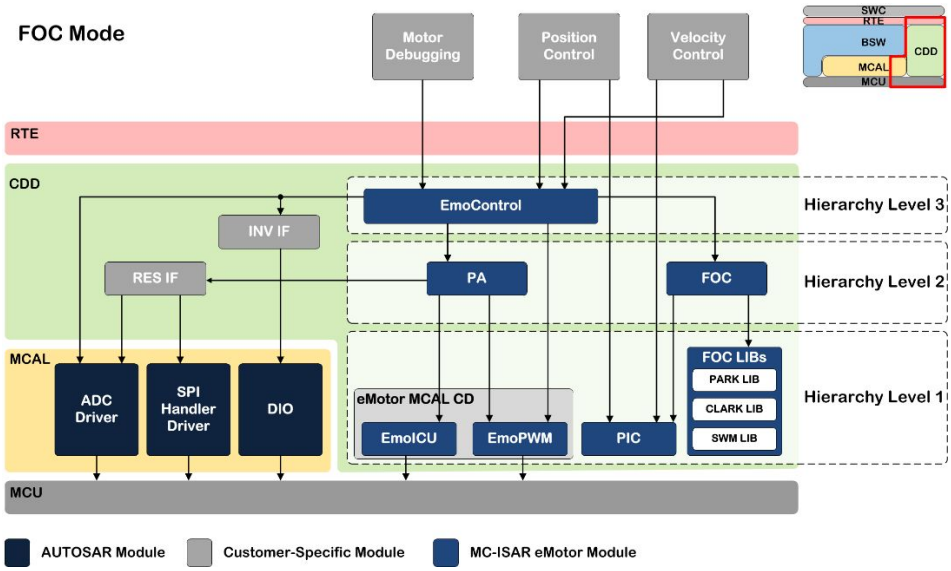


그림 3.6 FOC Mode 의 모듈 구성도.

Fig. 3.6 The Module Block diagram of FOC Mode.

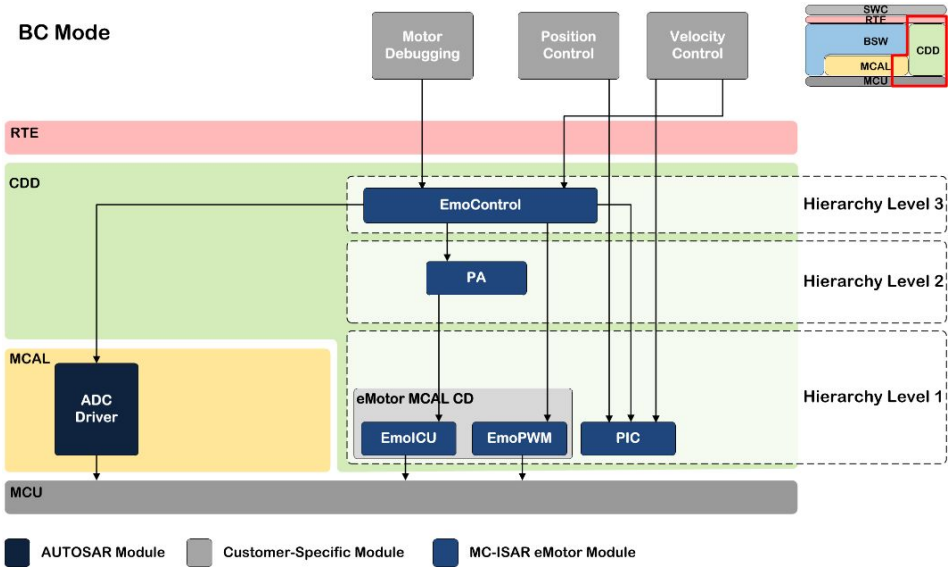


그림 3.7 BC Mode 의 모듈 구성도.

Fig. 3.7 The Module Block diagram of BC Mode.

BLDC 모터 구동을 위해 전기적 정류 방법인 BC Mode 를 사용한다. BC Mode 는 그림 3.7 과 같이 Hierarchy Level 3 의 EmoControl 모듈, 위치 검출을 위한 Hierarchy Level 2 의 PA 모듈, Hierarchy Level 1 의 속도 검출을 위한 EmoIcu, PWM 생성을 위한 EmoPWM, 제어기 구현을 위한 PI 제어기 모듈을 사용하고, MCAL 계층에서 제공하는 ADC 모듈을 사용한다. Hierarchy 배치에 따른 API 사용에 관한 설명과 구현한 BC Mode 에 대한 실험환경 구축 및 방법은 4 장에서 설명한다.

4. 실험환경 구축 및 실험 결과

4.1 eMotor Driver 기반 모터 제어기 구성

본 논문에서 구현하는 eMotor Driver 기반 BLDC 모터 제어시스템의 구성도는 그림 4.1 과 같이 MCU, 인버터, BLDC 모터 및 홀센서로 구성되어있다. MCU 는 Infineon 사의 32bit MCU 인 TC1798 을 사용하였다. MCU 는 전자적인 정류제어 방법인 BC Mode 가 구현되었고, BLDC 모터에 3 상의 전류를 인가해주기 위한 인버터 및 모터의 회전자 위치를 검출하기 위한 홀센서가 있다. eMotor Driver 에서 설정해준 TC1798 의 모듈은 CCU6(Capture/Compare Unit 6), ADC 이며, CCU6 는 PWM 신호를 스위칭 해주는 CC6x, COUT6x 와 Hall Pattern 을 받아오는 CCPOSx (x = 0, 1, 2) 레지스터가 사용된다. ADC 는 BC Mode 를 동작시키기 위해 사용한다.

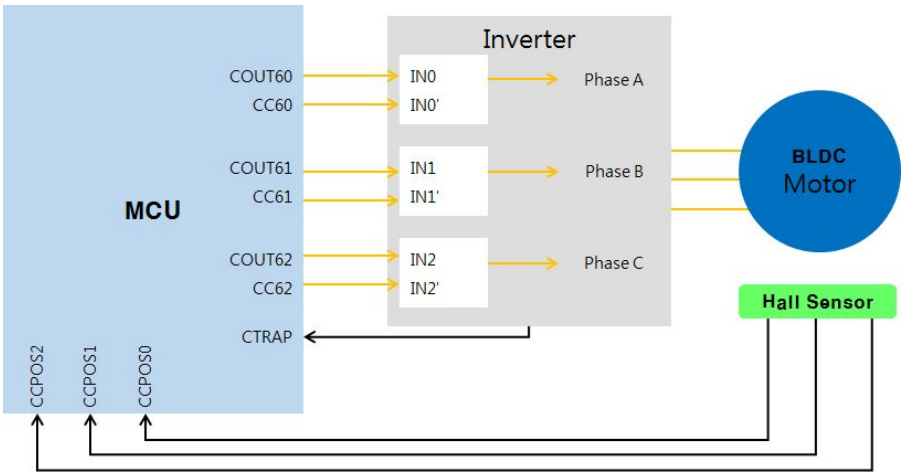


그림 4.1 eMotor Driver 를 적용한 BLDC 모터 제어시스템.

Fig. 4.1 The BLDC Motor Control System based eMotor Driver.

BLDC 모터 구동을 위해서는 eMotor Driver 의 BC Mode 를 이용하며, 그림 3.7 과 같이 BC Mode 에서 선정된 모듈을 기준으로 사용되는 함수의 흐름도는 그림 4.2 와 같다.

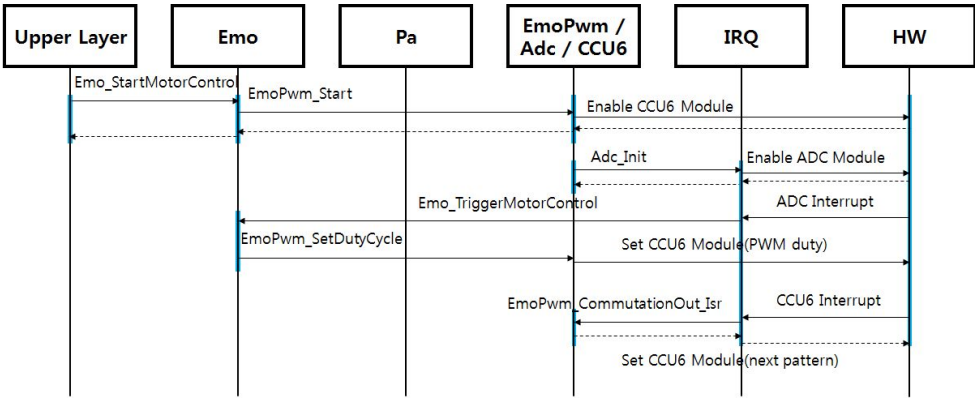


그림 4.2 BC Mode 기반 BLDC 모터 구동 흐름도.

Fig. 4.2 The Flowchart of BLDC Motor based BC Mode.

가장 최상위의 Layer 에서 모터 구동을 위해 Emo_StartMotorContol API 를 호출하면 Emo.c 파일에서 EmoPwm_Start API 를 호출하여 CCU6 모듈을 활성화하고, CCU6 레지스터에서 처음의 Hall Pattern 을 받아온다. BC Mode 알고리즘을 실행시키기 위해서는 Emo_TriggerMotorControl API 를 동작시켜야 하고, 해당 API 는 ADC 모듈의 인터럽트에 의해 동작하므로 ADC_Init API 를 통해 ADC 모듈을 활성화한다. Emo_TriggerMotorControl API 가 ADC 인터럽트에 의해 호출되면, CCU6 에 PWM duty 를 설정하기 위해 EmoPwm_SetDutyCycle API 를 호출하여 CCU6 레지스터에 duty 값을 저장해 둔다.

모터가 회전하면 회전자 위치에 따라 Hall Pattern 값이 바뀌게 되고 하드웨어적으로 측정한 Hall Pattern 이 설정해놓은 다음의 Hall Pattern(Next Required Hall Pattern)과 일치할 때 CCU6 의 인터럽트가 발생한다. 해당 CCU6 인터럽트는 EmoPwm_CommutationOutput_Isr API 를 호출하고, 현재 Hall Pattern 에 따라 요구되는 다음의 Hall Pattern(Next Required Hall Pattern)을 불러와서 지속적으로 Hall Pattern 을 갱신한다.

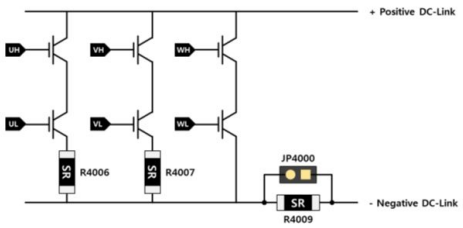
BLDC 모터 구동을 위해서는 2.2 절에 명시한 바와 같이 Hall Pattern 과 PWM Pattern 이 필요하다. 본 논문의 실험에서 사용되는 SyncWork 사의 BLDC 모터의 사양은 표 4.1 와 같고, SyncWork 사의 인버터 보드는 그림 4.3 와 같은 Hall

Pattern 과 PWM Pattern 을 필요로 한다. 이는 2 장 2.2 절에 BLDC 모터의 구동 원리의 표 2.1 과 동일하다[18].

표 4.1 BLDC 모터 사양

Table 4.1 The specifications of BLDC Motor

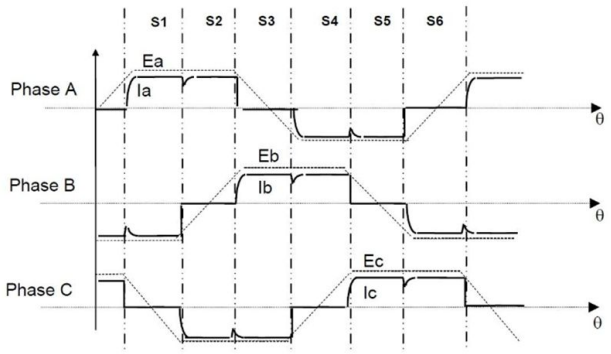
구분	값	단위
최대 전압	24	V
최대 전류	1.1	A
전력	15 (~28)	W
최대 속도	4580	RPM
토크	0.03	N.m
극수	8	
상(Phase) 저항	4.5	Ohm
인덕턴스	1.2	mH



(a) 인버터 점퍼 핀-헤더의 이해

State	S1	S2	S3	S4	S5	S6
WL	0	1	1	0	0	0
WH	0	0	0	0	1	1
VL	1	0	0	0	0	1
VH	0	0	1	1	0	0
UL	0	0	0	1	1	0
UH	1	1	0	0	0	0
Ia	+Ve	+Ve	0	-Ve	-Ve	0
Ib	-Ve	0	+Ve	+Ve	0	-Ve
Ic	0	-Ve	-Ve	0	+Ve	+Ve

(b) BLDC의 Commutation State



(c) BLDC의 상전류 및 역기전력 파형

그림 4.3 3상 BLDC 모터 구동 예상 결과.

Fig. 4.3 The Expected Result of driving BLDC Motor.

Hall, PWM Pattern 값은 그림 4.4 와 같이 MCU 의 CCU6 모듈의 MCMOUTS(Multi-Channel Mode Output Shadow Register)값에 저장 된다. 홀 센서로부터 입력된 현재의 Hall Pattern 은 MCMOUTS 레지스터의 CURHS(Current Hall Pattern Shadow) 필드에 저장된다. 일정한 토크를 생성해 주기 위해 예측되는 다음의 Hall Pattern 은 MCMOUTS 레지스터의 EXPHS(Expected Hall Pattern Shadow)

필드에 저장된다. 현재 Hall Pattern 에 따른 PWM Pattern 은 MCMPS(Multi-Channel PWM Pattern) 필드에 저장된다[19].

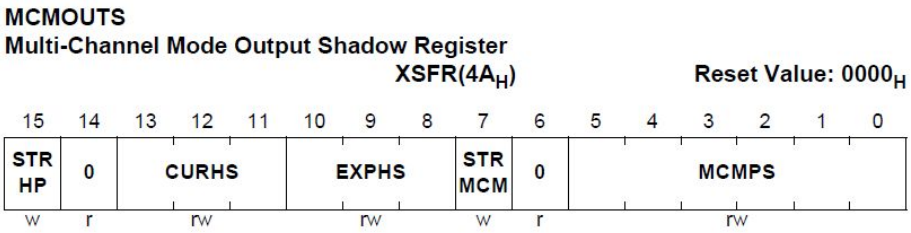


그림 4.4 CCU6 모듈의 MCMOUTS 레지스터.

Fig. 4.4 MCMOUTS Register of CCU6 Module.

4.2 실험환경 구축

실험에 사용된 프로그램은 AUTOSAR 계층 중 SWC 개발 부분을 제외한 RTE 를 포함하고 하위 BSW 모듈들에 대한 configuration 및 generation 을 제공하는 EB tresos Configuration Tool 을 사용하고 이는 그림 4.5 와 같다.

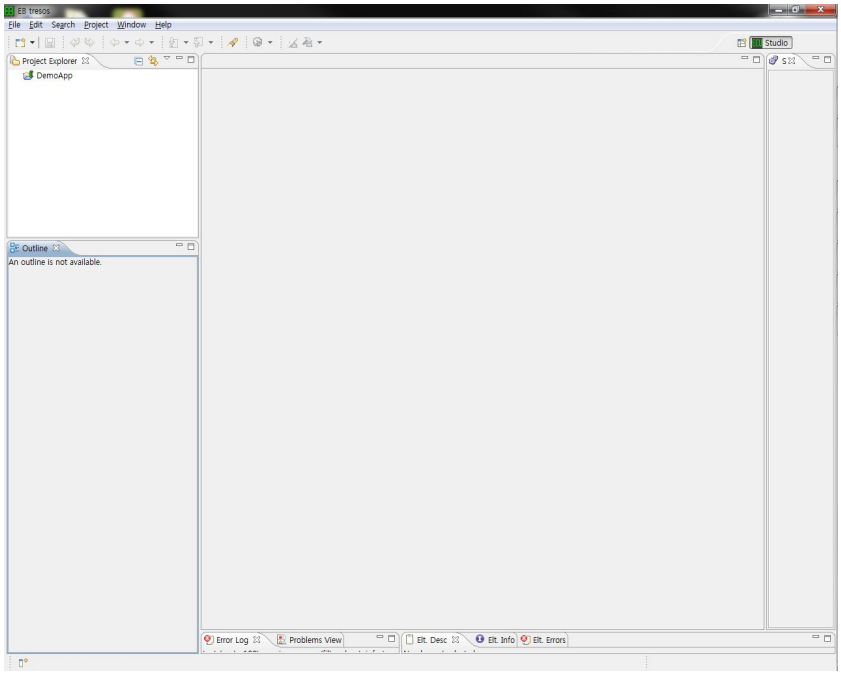


그림 4.5 EBtresos tool.

Fig. 4.5 EBtresos Tool.

EB tresos 를 사용하여 프로젝트를 생성하여 Target Board 와 사용할 모듈을 그림 4.6 과 같이 설정한다. 또한 BC Mode, Hall Pattern 과 PWM Pattern, 모터 사양 등을 입력 및 설정하고, code generation 을 통해 C 파일과 헤더 파일을 생성한다. 생성된 파일은 TASKING 을 통해 컴파일 한다. 본 절에서는 각 모듈 설정을 통해 실험환경을 구축하는 과정을 살펴본다.

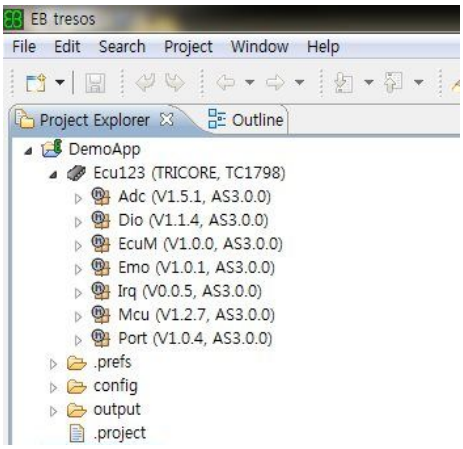


그림 4.6 EBtresos tool 을 이용한 실험환경 구축.

Fig. 4.6 the Construction of experimental environment using EBtresos.

EB tresos 를 통해 프로젝트 생성 후 그림 4.6 과 같이 MCU Target(TC1798)과 사용할 모듈을 선정한다. 여기서 사용되는 모듈은 Adc, Dio, EcuM, Emo, Irq, Mcu, Port 가 있다. Emo 모듈은 eMotor Driver 를 통해 3 상 모터를 구동하기 위한 대표적인 모듈이며, Emo 모듈을 생성해주면 그림 4.7 과 같이 2 개의 탭(EmoConfigSet, EmoGeneral)이 나타난다.

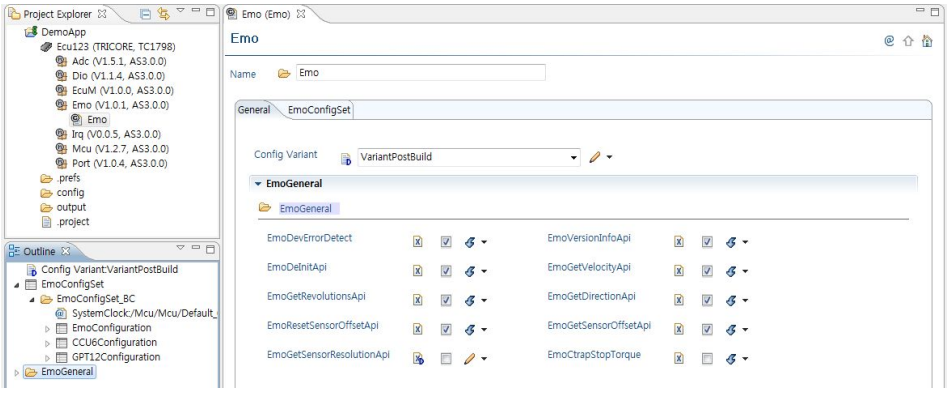


그림 4.7 Emo 모듈의 API.

Fig. 4.7 the API of Emo Module.

Emo 모듈에서는 그림 4.6 과 같이 EmoGeneral 을 통해 API 를 선택할 수 있다. 특정 API 를 선택하면 API 에 대한 소스코드가 생성되는 것이 아니라, 선택한 API 에 따른 소스코드를 사용할 수 있게 활성화 된다. 사용자는 활성화된 소스코드를 조합하여 모터 구동에 필요한 최상위 Hierarchy Level 3 를 작성할 수 있다. EmoConfigSet 에 EmoCongifSet_BC 모듈을 추가해주면 3 개의 탭(EmoConfiguration, CCUConfiguration, GPT12Configuration)이 나타난다. System Clock 탭에서 그림 4.8 과 같이 Emo 모듈의 System Clock 을 설정해준다. System Clock 은 MCU 모듈에서 설정한 Clock Reference 와 동일하게 사용한다.

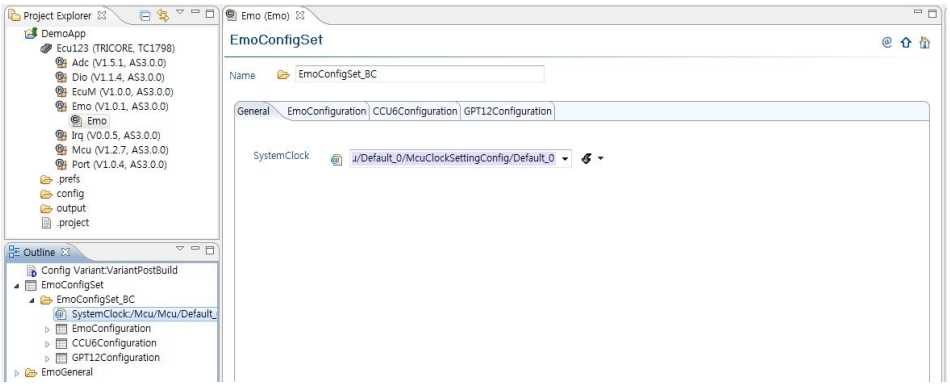


그림 4.8 Emo 모듈의 시스템 클럭 설정.

Fig. 4.8 the Setting of System Clock for Emo Module.

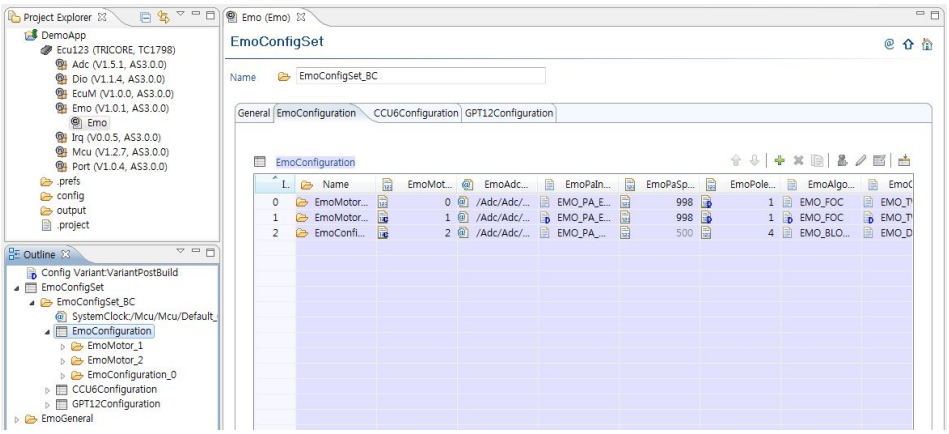


그림 4.9 Emo 모듈의 기본 사항 설정.

Fig. 4.9 the Setting of Basic Data for Emo Module.

그림 4.9 와 같이 EmoConfigSet_BC 에서 EmoConfiguration 에 들어가면 모터를 추가할 수 있고, 최대 4 개까지 가능하다. 여기서는 기본 사항(EmoMotor Id, EmoAdcGroupRef)을 그림 4.10 과 같이 설정할 수 있고, 회전자 위치 검출을 위해 사용하는 센서와 BC Mode 등을 설정할 수 있다.

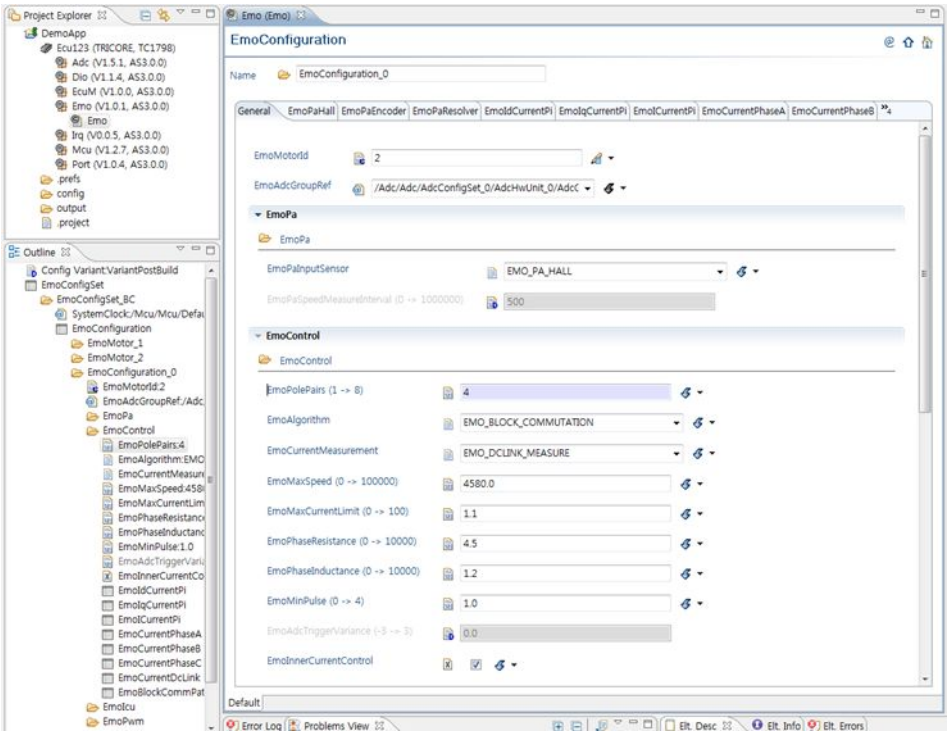


그림 4.10 Emo 모듈의 BC Mode 설정.

Fig. 4.10 the Setting of BC Mode for Emo Module.

EB tresos 상에서 BLDC 모터 구동을 위해 생성한 EmocConfiguration 모듈은 EmoConfiguration_0 이다. 해당 모듈을 생성하면 EmoPa, EmoControl, EmoIcu, EmoPWM 탭이 나타나고, EmoPa 에서는 엔코더, 레졸버, 홀 센서, 센서리스 중 사용하고자 하는 센서를 선택하면 된다. BLDC 모터 구동을 위해 사용된 Pa 센서는 홀 센서이므로 홀센서를 선택한다. EmoControl 에서는 모터 사양에 맞는 파라미터를 입력하고, BC Mode 혹은 FOC Mode 를 선택할 수 있다. BLDC 모터 구동을 위해 사용한 EmoAlgorithm 은 BC Mode 이다. EmoCurrentMeasurement 는

DC Link Measurement 방식을 사용한다. BC Mode 는 전류 측정을 위해 반드시 DC Link Measurement 방식을 사용해야 한다. 이러한 이유는 그림 4.11 과 같이 [16]에 명시되어 있는 것을 기준으로 선정하고, 이는 사용자가 임의로 전류 측정방식을 설정할 수 없다. BC Mode 에서 사용하는 DC Link Measurement 방식은 그림 4.11 과 같은 특징을 가지며, 이를 준수하여 Configuration Tool 을 다루어야 한다[16].

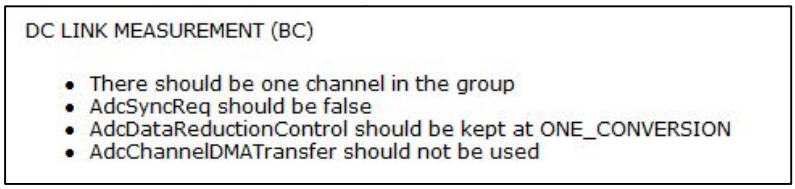


그림 4.11 BC Mode 설정을 위한 준수사항(1/2).

Fig. 4.11 the Compliance details for the configuration of BC Mode(1/2).

BLDC 모터 사양에 맞게 EmoPolePairs, EmoMaxSpeed 등을 입력하고, EmoInnerCurrentControl 을 선택한다. 이 Mode 는 그림 4.12 를 준수하여 설정한 것으로[16], EmoInnerCurrentControl 이 활성화 되어야 Hall Pattern 과 PWM Pattern 이 계속 갱신된다.

Name	EmoInnerCurrentControl
Type	Boolean
Range	false,true
Default	false
Configuration Class	PostBuild
Description	false: block commutation run the motor with speed control only true: block commutation run the motor with inner current control
Dependency	EmoAlgorithm = EMO_BLOCKCOMMUTATION

그림 4.12 BC Mode 설정을 위한 준수사항(2/2).

Fig. 4.12 the Compliance details for the configuration of BC Mode(2/2).

EmoConfiguration_0 에서 Hall Pattern 과 PWM Pattern 을 설정하는 것은 그림 4.13 과 같이 EmoControl 에 EmoBlockCommPattern 에서 입력한다. BC Mode 에 필요한 Hall Pattern 과 PWM Pattern 을 그림 4.14 와 같이 입력한다.

EmoBcPattern 파라미터를 그림 4.4 에 CURHS, EXPHS, MCMPS 순서대로 그림 4.14 와 같이 입력한다. 입력한 값들은 표 2.1 의 Hall Value, Next Required Hall Value, PWM 과 동일하게 입력한다.

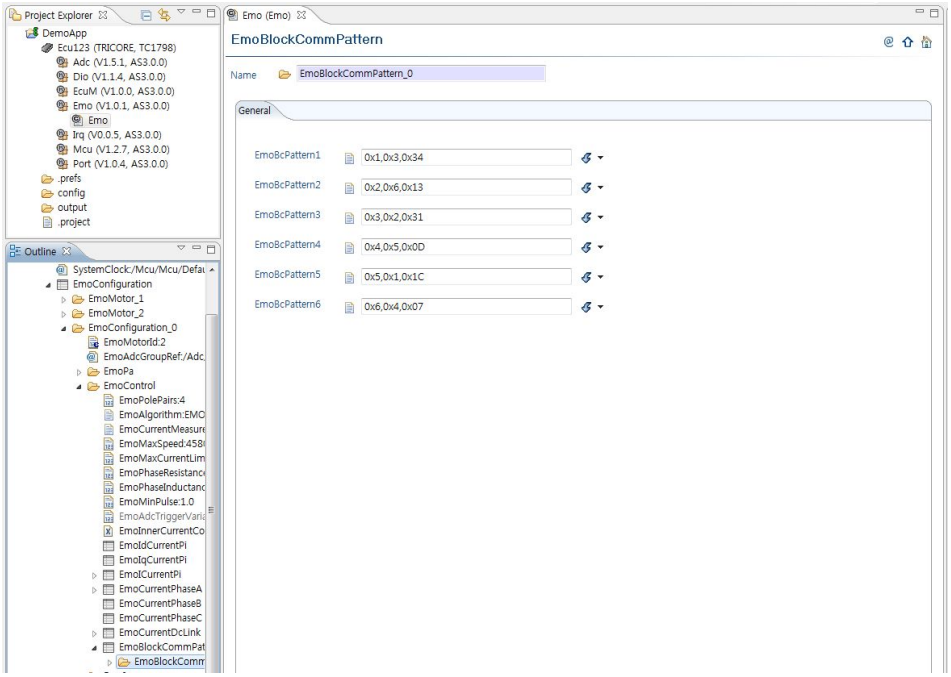


그림 4.13 Hall Pattern 및 PWM Pattern 설정.

Fig. 4.13 the Setting of Hall Pattern and PWM Pattern.

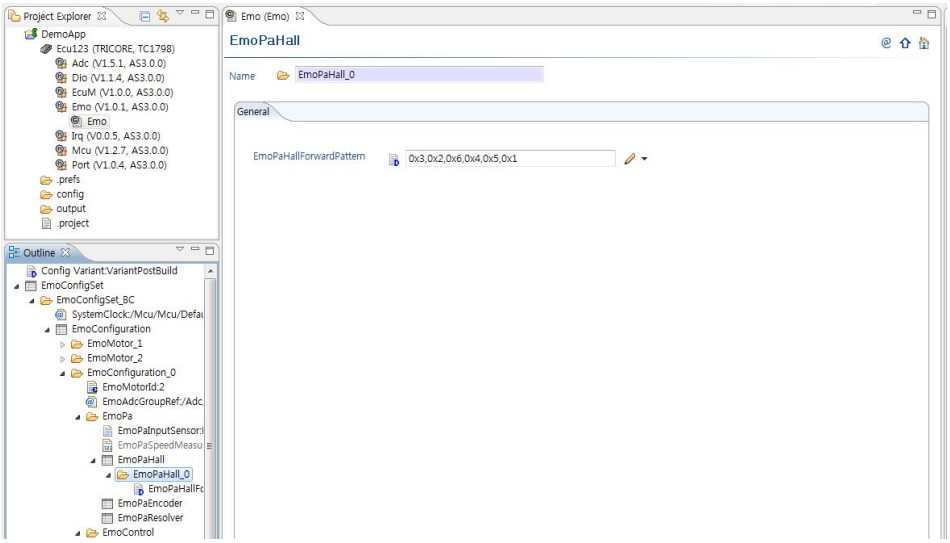


그림 4.14 Hall Pattern 방향 설정.

Fig. 4.14 the Setting of Hall Pattern Direction.

EmoControl 에서 Hall Pattern 과 PWM Pattern 을 설정하고, 모터가 시계방향 혹은 반 시계방향으로 회전하기 위해서는 Hall Pattern 의 방향성을 결정해주어야 한다. 표 2.1 과 그림 4.3 의 정류자 상태를 토대로 반 시계방향 구동을 위해 EmoPa 에서 EmoPaHall 의 EmoPaHallForwadPattern (3→2→6→4→5→1)을 그림 4.14 와 같이 설정한다.

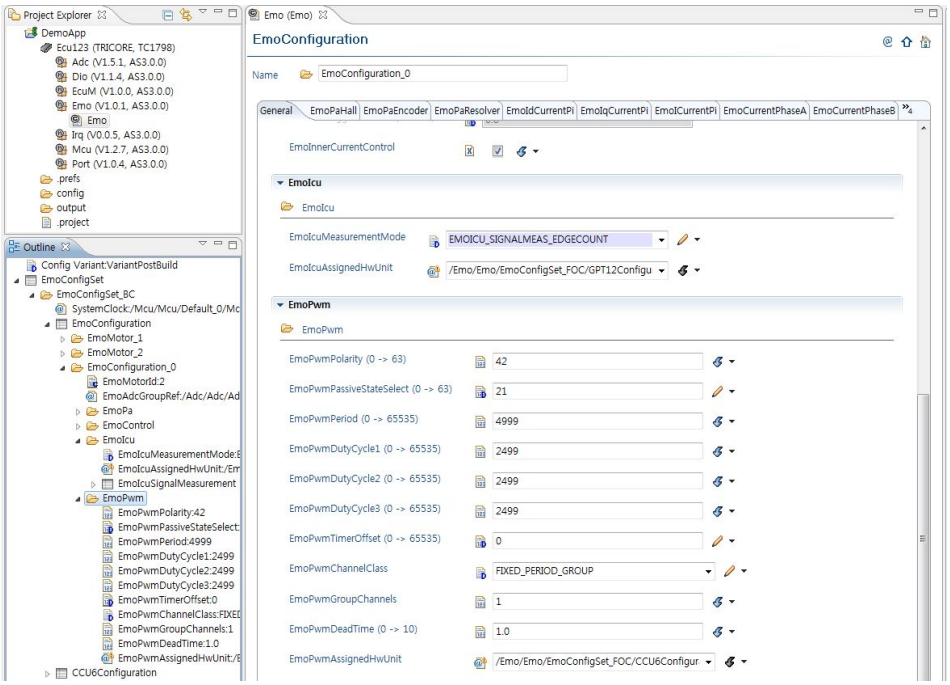


그림 4.15 Emo 모듈의 EmoPWM 및 EmoIcu 설정.

Fig. 4.15 the Setting of EmoPWM and EmoIcu for Emo Module.

Emo 모듈에서 속도 검출을 위한 EmoIcu, GPT12(General Purposed Timer), 인버터 스위칭과 관련된 EmoPwm, CCU6 설정은 그림 4.15 와 같다. 특히 EmoPWM 모듈에서 그림 4.16 과 같이 EmoPwmPloarity, EmoPwmPassiveStateSelect, EmoPwmPeriod, EmoPwmDutyCycle 파라 미터들은 값 설정에 따라 출력되는 PWM 에 영향을 준다. EmoPwmPolarity 와 EmoPwmPassiveStateSelect 는 인버터 사양, CCU6 의 Passive State Level Register 레지스터를 고려하여 이에 맞는 값을 설정한다. EmoPwmPeriod 는 통상적으로 20 kHz (50 μs)를 사용하므로 이에 해당하는 Period Register 값인 4999(0x1387)가 입력되고, 50%의 duty cycle 을 주기 위한 Period Register 에는 2499(0x09C3)이 입력된다.

Name	EmoPwmPolarity
Type	Integer
Range	0 -> 63
Default	42
Configuration Class	PostBuild
Description	The polarity of 6 PWM channels, with the LSB being CC60 (IL1), followed by COUT60 (IH1), CC61 (IL2), COUT61(IH2), CC62(IL3), COUT62(IH3).
Dependency	None

Name	EmoPwmPassiveStateSelect
Type	Integer
Range	0 -> 63
Default	21
Configuration Class	PostBuild
Description	The passive state of 6 PWM channels, with the LSB being CC60 (IL1), followed by COUT60 (IH1), CC61 (IL2), COUT61(IH2),CC62(IL3), COUT62(IH3).
Dependency	None

Name	EmoPwmPeriod
Type	Integer
Range	0 -> 65535
Default	3124
Configuration Class	PostBuild
Description	Period of the PWM channel / emotor. Typically 20 KHz.
Dependency	None

Name	EmoPwmDutyCycle1
Type	Integer
Range	0 -> 65535
Default	100
Configuration Class	PostBuild
Description	Default duty cycle of the first channel.
Dependency	None

그림 4.16 EmoPwm 모듈의 파라미터 설정.

Fig. 4.16 the Setting parameters of EmoPwm Module.

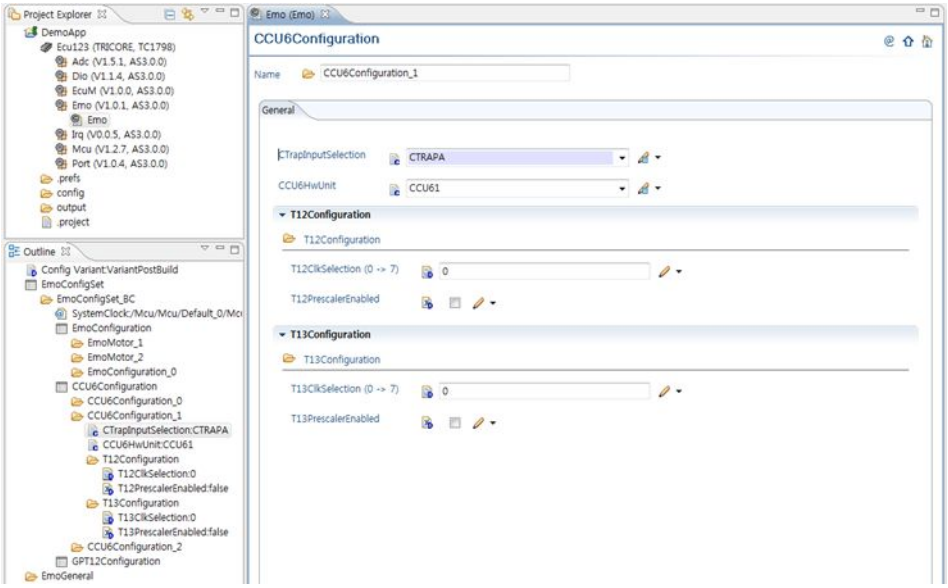


그림 4.17 CCU6 모듈의 기본 사항 설정.

Fig. 4.17 the Setting of Basic Data for CCU6 Module.

Emo 모듈에서 CCU6 Configuration 탭은 그림 4.17 과 같이 사용하는 CCU6 의 하드웨어를 선택할 수 있다. 본 논문의 실험에서는 CCU61 을 사용하고, 인버터에 과전류를 감지하여 PWM 신호를 차단해 주는 외부의 시그날인 CTRAPA 를 사용한다.

ADC 모듈은 Emo 모듈의 BC Mode 를 동작시키는 중요한 요소이며, 그림 4.10 상단에 EmoAdcGroupRef 를 통해 ADC 모듈, 그룹을 설정해 줄 수 있다. ADC 모듈에서 AdcHwUnit 을 선택하고, ADC 모듈의 하드웨어 사항, Request Source 의 우선순위 등을 그림 4.18 과 같이 설정한다.

AdcHwUnit_0 을 생성하면 AdcChannel, AdcGroup, AdcInputClass 등이 나타난다. AdcChannel 에서 그림 4.11 에 준수하여 그림 4.18 과 같이 설정한다.

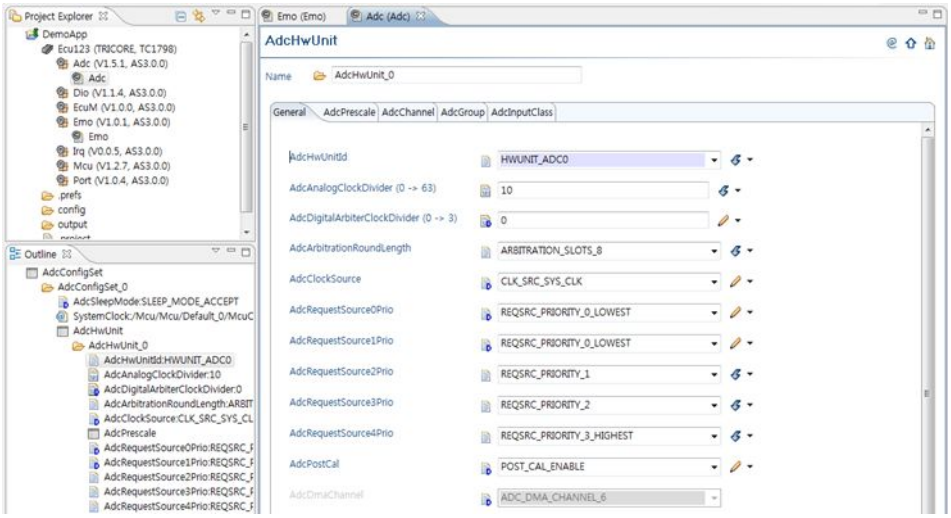


그림 4.18 ADC 모듈의 기본 사항 설정(1/2).

Fig. 4.18 the Setting of Basic Data for ADC Module(1/2).

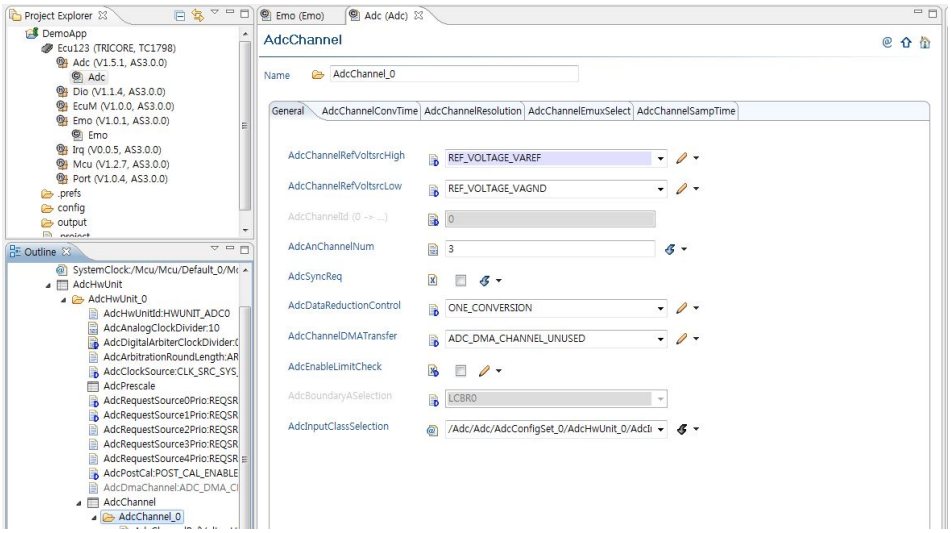


그림 4.19 ADC 모듈의 기본 사항 설정(2/2).

Fig. 4.19 the Setting of Basic Data for ADC Module(2/2).

AdcHwUnit_0 에 AdcGroup 에서 AdcGroup_BC 를 생성하고 그림 4.20 과 같이 설정한다. AdcGroup_BC 에서 AdcGroupTriggSrc 는 ADC 그룹의 트리거 타입을 설정하는 부분으로, SW API 로부터 트리거 할 것인지, HW 로부터 트리거 할

것인지 결정하는 부분이며 본 실험에서는 CCU6 를 통해 ADC 모듈의 그룹을 HW 적으로 트리거한다.

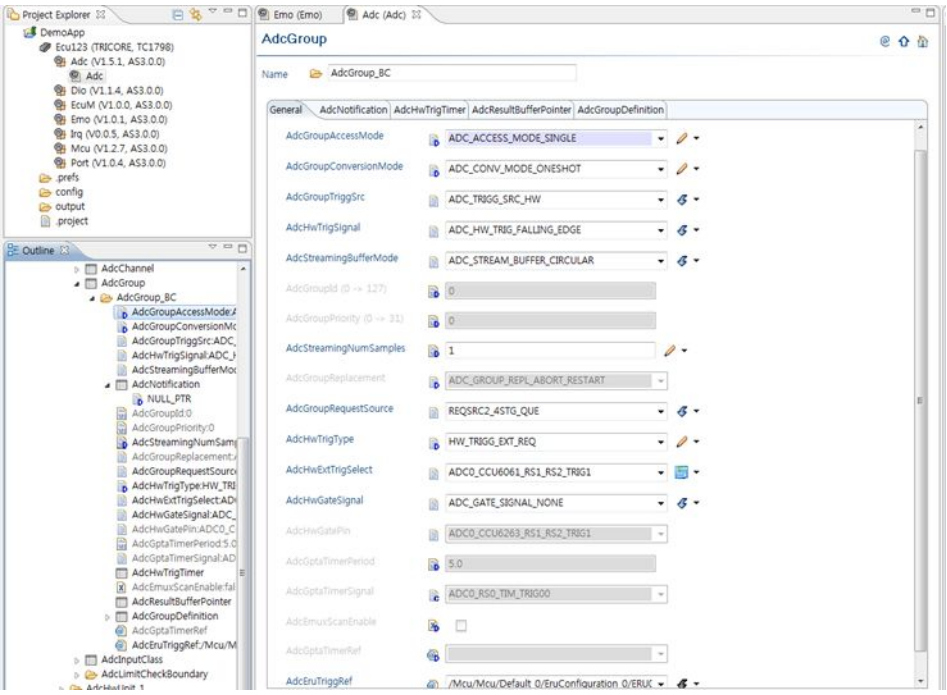


그림 4.20 ADC 모듈의 기본 사항 설정(2/2).

Fig. 4.20 the Setting of Basic Data for ADC Module(2/2).

CCU Module	Trigger Line Value	ADC Request Source
0	CCU60_COUT63 (Trigger line 2, Trigger Sel 0)	Request Source 4
1	CCU61_COUT63 (Trigger line 1, Trigger Sel 1)	Request Source 2
2	CCU62_COUT63 (Trigger line 0, Trigger Sel 0)	Request Source 0
3	CCU63_COUT63 (Trigger line 1, Trigger Sel 1)	Request Source 1

그림 4.21 ADC 트리거를 위한 준수 사항.

Fig. 4.21 the Compliance details for the configuration of ADC Trigger.

ADC 그룹 트리거 시 준수 사항은 그림 4.21 과 같고, CCU61 모듈을 사용하므로 Trigger Line Value 는 CCU61_COUT63(Trigger Line1, Trigger Sel1), ADC Request Source 는 2 를 사용한다. 이를 AdcGroup_BC 에 적용하면

AdcGroupReauest Source 는 REQSRC2(Request Source 2)를, AdcHwExtTrigSelect 는 ADC0_CCU6061_ RS1_RS2_TRIG1(ADC0_CCU61_Trigger Line1_Trigger Line2_Trigger Sel1)을 선택한다[16].

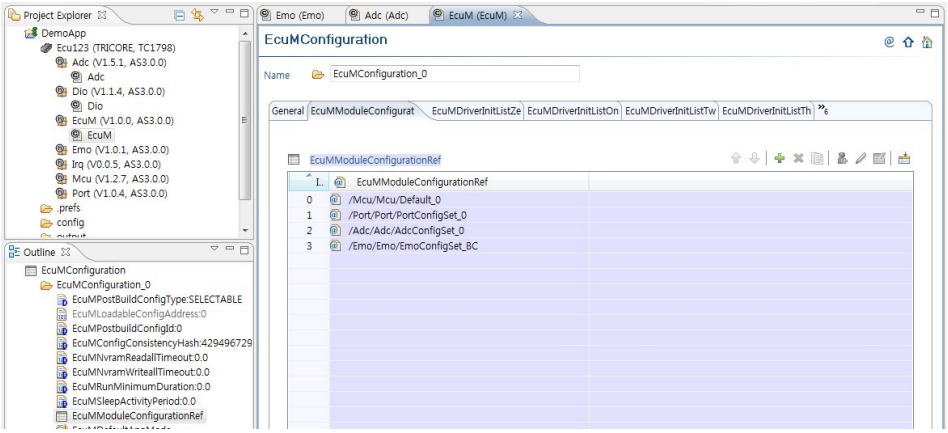


그림 4.22 EcuM 모듈의 기본 사항 설정(1/2).

Fig. 4.22 the Setting of Basic Data for EcuM Module(1/2).

EB tresos 상에서 EcuM 모듈은 ECU State Manager 로 그림 4.22 와 같이 EcuMConfigurationRef 에서 선택한 다른 모듈(Mcu, Port, Adc, Emo)의 Init function(초기화 함수)를 설정한다. 여기서 Init 함수는 EcuM 모듈에서 제공하는 서비스 함수이며, 다른 모듈들은 EcuM 모듈에서 EcuMModuleId 로 나뉘게 된다. 이러한 설정은 EcuM 모듈에서 생성해준 EcuMConfiguration 에 EcuMDriverInitListOne 에서 설정할 수 있고 그림 4.23 과 같다.

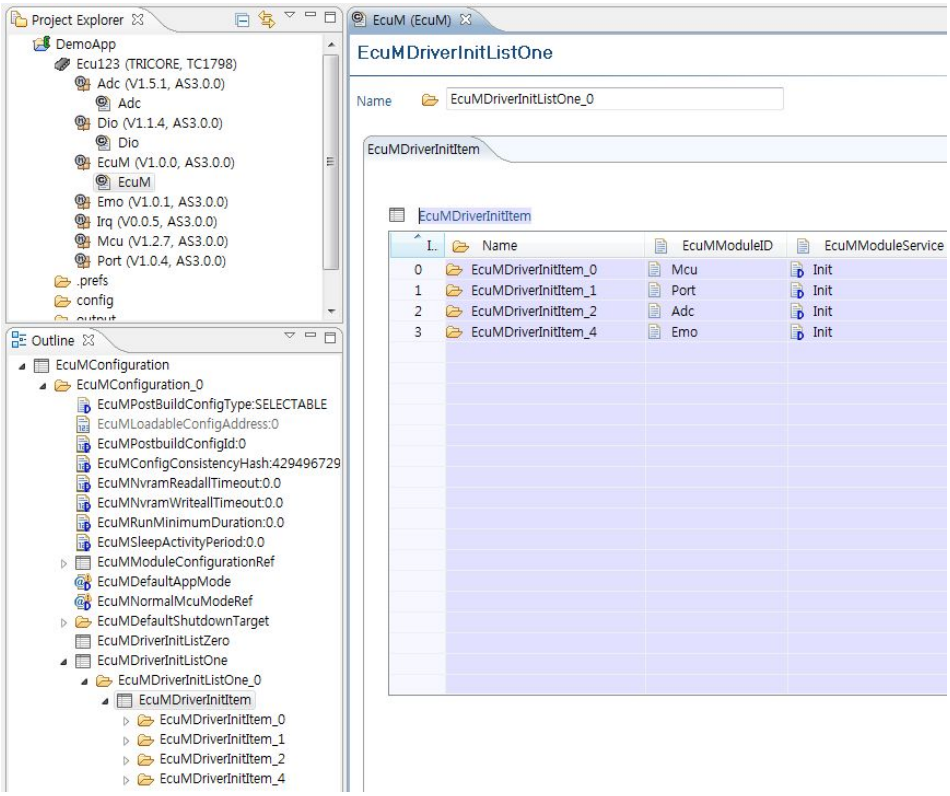


그림 4.23 EcuM 모듈의 기본 사항 설정(2/2).

Fig. 4.23 the Setting of Basic Data for EcuM Module(2/2).

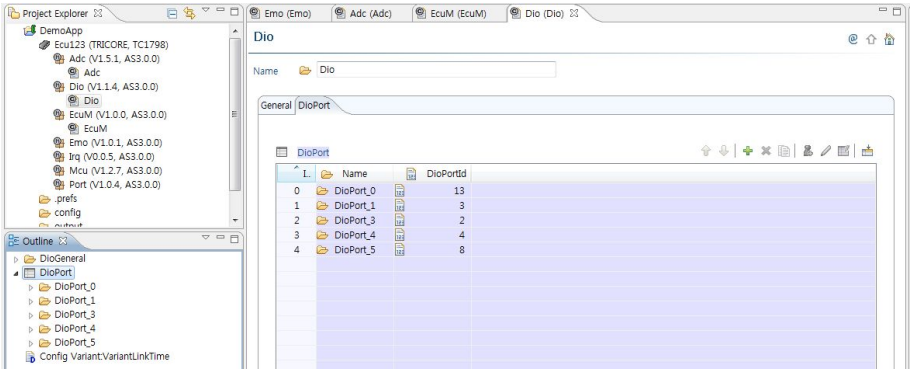


그림 4.24 Dio 모듈의 기본 사항 설정(1/2).

Fig. 4.24 the Setting of Basic Data for Dio Module(1/2).

EB tresos 상에서 Dio 모듈은 Digital Input Output 모듈로서 DIO 채널을 개별적으로 설정해줄 수 있는 부분이며 그림 4.24 와 같고, 하드웨어의 특정 채널의 채널을 DioPort 에서 생성할 수 있다.

PinningTC1798 Pin Configuration

Table 2 Pin Definitions and Functions (PG-LFBGA- 516 Package) (cont'd)

Pin	Symbol	Ctrl.	Type	Function
J13	P2.10	I/O	A1/ PU	**Port 2 General Purpose I/O Line 10**
	IN2	I		**IN2 Line of GPTA0**
	IN2	I		**IN2 Line of GPTA1**
	IN2	I		**IN2 Line of LTCA2**
	T12HRE	I		**CCU60**
	CC61INC	I		**CCU60**
	CTRAPA	I		**CCU61**
	CC60INC	I		**CCU61**
	CTRAPB	I		**CCU63**

그림 4.25 Dio 모듈을 위한 준수 사항.

Fig. 4.25 the Compliance details for the configuration of Dio Module.

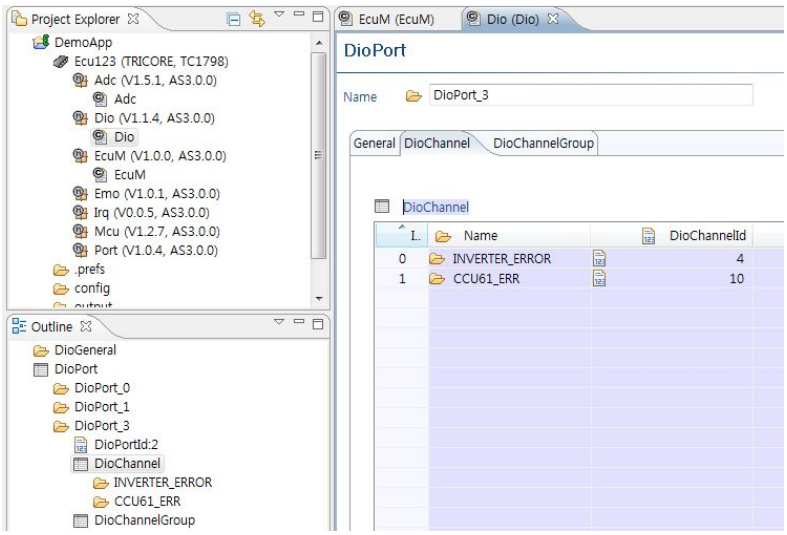


그림 4.26 Dio 모듈의 기본 사항 설정(2/2).

Fig. 4.26 the Setting of Basic Data for Dio Module(2/2).

그림 4.17 에서 CCU61 의 CTRAPA 를 사용하기 위해 설정하였고, 실제 CTRAPA 채널을 활성화 시켜주기 위한 부분이 Dio 모듈이다. TC1798 의 데이터 시트에서 CCU61/CTRAPA 를 찾아보면 2.10 포트인(2 번 포트 10 번 라인) 것을 그림 4.25 와

같이 확인할 수 있다[19]. Dio 모듈에서 DioPort 를 생성해주고 DioPortid 를 2 번으로 설정하면 Port 2 가 활성화된다. DioPort 안에 DioChannel 에서는 2 번 포트에 몇 번 라인을 활성화 해줄 것인지 결정할 수 있고, 이름은 편의상 CCU61_ERR 로 정하였다. Port 2 에서 CCU61_ERR 는 10 번 라인을 사용하므로 DioChannelId 는 10 으로 정하고 이는 그림 4.26 과 같다. 따라서 Dio 모듈에서 CCU61 의 CTARPA 를 활성화 해주는 2.10 포트를 활성화 했다.

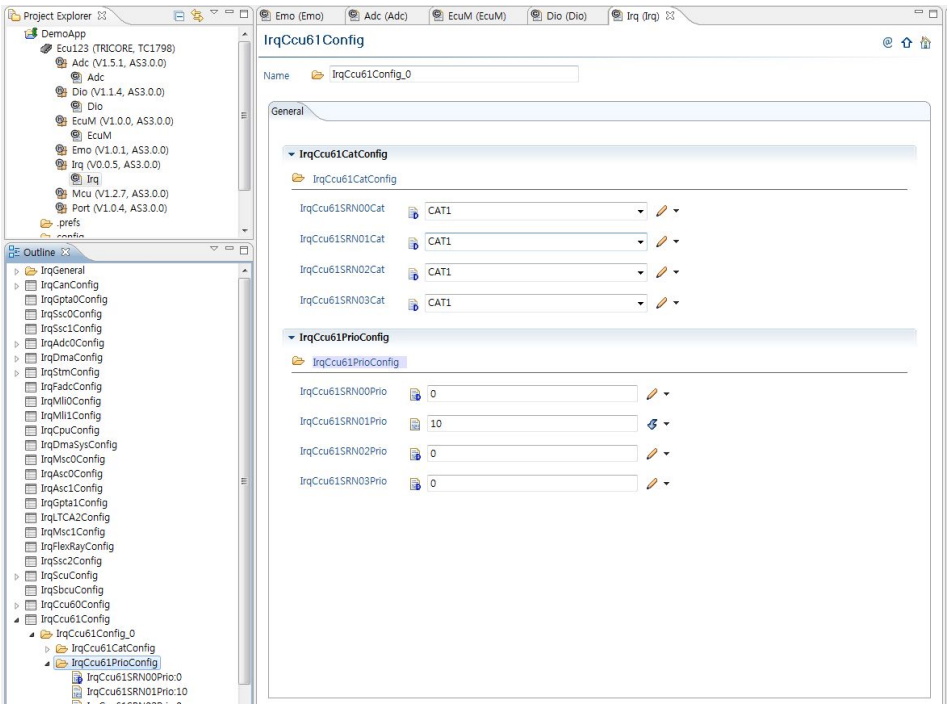


그림 4.27 Irq 모듈의 기본 사항 설정.

Fig. 4.27 the Setting of Basic Data for Irq Module.

다음으로 Irq 모듈은 하드웨어의 인터럽트의 우선순위와 인터럽트 카테고리를 설정하는 부분으로 그림 4.27 과 같이 MCU 내 존재하는 모듈 중 인터럽트를 발생시킬 수 있는 모듈이 모두 나열되어있다. 본 논문에서 사용한 CCU61 의 인터럽트 우선순위를 설정해 주기 위해 IrqCcu61Config 모듈에서 IrqCcu61Config_0 을 생성해주면 그림 4.27 과 같이 인터럽트 카테고리(IrqCcu61CatConfig)와 우선순위(IrqCcu61PrioConfig)를 설정해 줄 수 있다.

카테고리 1(CAT 1)을 사용하고, Source Request Number 1(IrqCc61SRN01Prio)을 사용할 것이기 때문에 우선순위를 가장 높게(10) 설정한다.

카테고리는 두 가지가 존재하는 데 CAT1 과 CAT23 이며, 이 개념은 OSEK/VDX (Open System and their interfaces for the Electronics in Motor Vehicles/Vehicle Distributed eXecutive) 라는 차량용으로 표준화된 OS 의 인터럽트 처리부분과 동일하다.

AUTOSAR 의 BSW 계층은 3.1 절에서 설명한 바와 같이 SWC 가 작업에 필요한 서비스를 제공하고 이 중에 하나가 OS 이다. OS 에는 대표적으로 OSEK OS(이하 OSEK)가 사용되고, OSEK 은 인터럽트 처리를 위해 ISR(Interrupt Service Routine)을 2 개의 카테고리로 나눈다. ISR 카테고리 1 은 ISR 내에서 운영체제 API 서비스를 사용하지 않고, 인터럽트가 종료되고 인터럽트가 태스크 관리에 영향을 주지 않을 경우, 인터럽트 처리를 계속 진행한다, 예를 들어 ISR 내에서 모터의 속도 검출만을 목적으로 한다면 ISR 카테고리 1 을 사용하면 된다. 반면에 ISR 카테고리 2 는 ISR 내에서 태스크 활성화 및 전환 등의 운영체제 API 서비스를 사용한다[20].

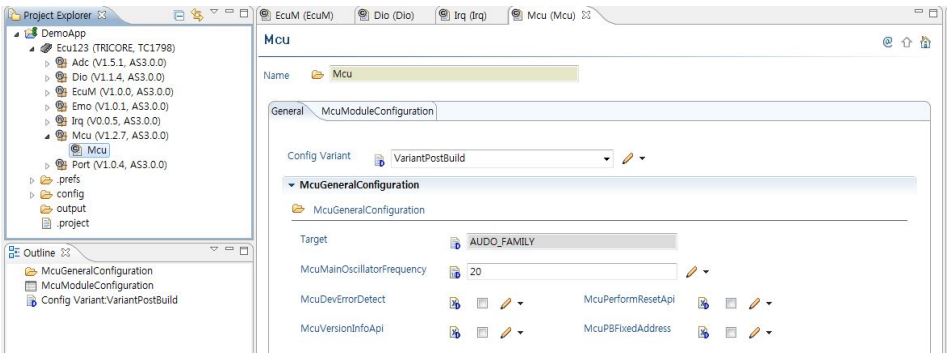


그림 4.28 Mcu 모듈의 기본 사항 설정.

Fig. 4.28 the Setting of Basic Data for Mcu Module.

마지막으로 Mcu 모듈은 그림 4.28 과 같이 사용할 하드웨어의 사양을 설정하고 타이머(GPTA: General Purposed Timer Array), 외부인터럽트(ERU: External Request Unit)를 활성화 해주는 부분이다. 이러한 설정 과정을 마치면

EB tresos 상에서 Code generation 버튼을 클릭하여, 사용자가 설정한 부분에 대한 C 파일과, H 파일이 생성된다.

본 절에서는 각 모듈 설정을 통해 실험환경을 구축하는 과정을 살펴보았으며, 생성된 파일들의 컴파일 과정은 [17]을 참고하여 실험을 진행하였다.

4.3 실험 결과

본 논문의 실험 결과는 BLDC 모터의 구동 특성을 파악할 수 있는 Hall Pattern, PWM Pattern, 상 전압을 통해 확인하고, PWM 의 duty 를 바꿀 때 모터 속도 변화에 대해 확인한다. 하이퍼터미널을 이용해 eMotor Driver 에서 제공하는 라이브러리에 쉽게 접근할 수 있도록 그림 4.29 와 같이 소스를 구현한다. 연결할 수 있는 모터 3(Control Motor 1, 2, 3)개 중 Control Motor 3 을 선택하면 BLDC 모터를 구동 혹은 정지시킬 수 있는 메뉴가 나타난다.

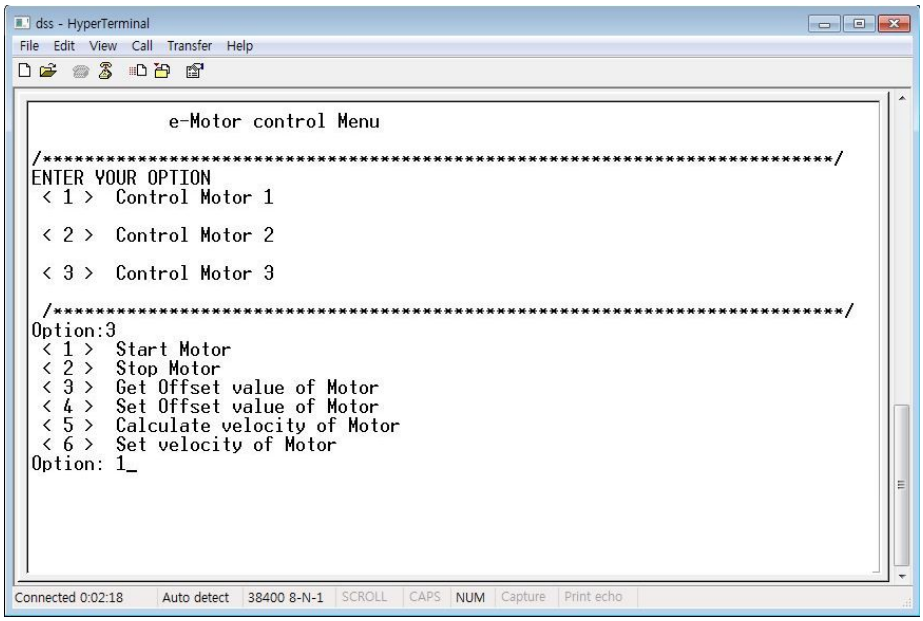


그림 4.29 BLDC 모터의 상위 Hierarchy Layer.

Fig. 4.29 the Upper Hierarchy Layer of BLDC Motor.

그림 4.30 은 3 개의 Hall 값 (Hall_U, Hall_V, Hall_W)과 모터에 인가되는 U, V, W 각 상의 전압을 측정한 것이며, 이는 2 절의 그림 2.5, 4 절의 그림 4.3 과 동일함을 확인할 수 있다. 또한 모터가 반 시계방향으로 회전함에 따라 출력되는 Hall Pattern 과 PWM Pattern 을 정리한 것은 표 4.2 와 같고, 이는 4 절의 그림 4.3 과 동일함을 확인할 수 있다.

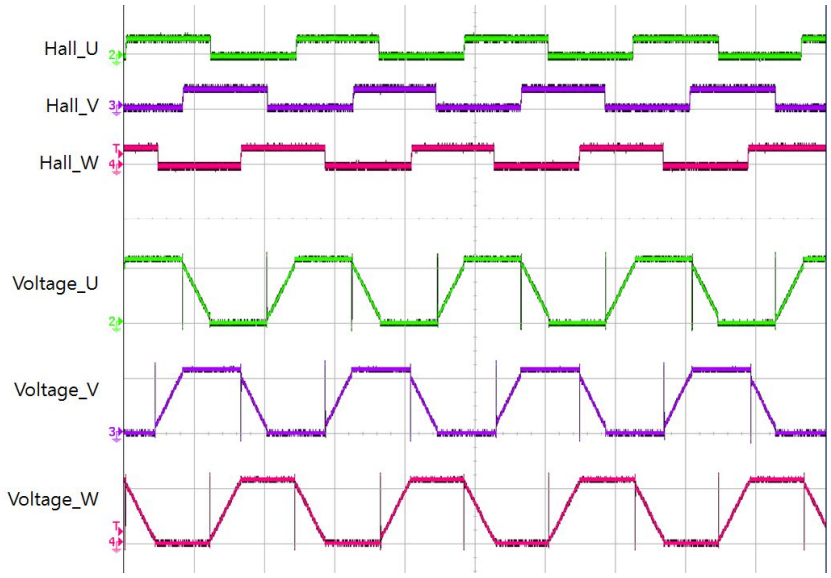


그림 4.30 BLDC 모터의 Hall Pattern과 3상 파형.

Fig. 4.30 the Result of Hall Pattern and 3 Phase with BLDC Motor.

표 4.2 BLDC 모터의 Hall Pattern과 PWM Pattern 결과

Table 4.2 the Result of Hall Pattern and PWM Pattern with BLDC Motor

Hall Pattern							
상위비트	Phase C	1	0	0	0	1	1
	Phase B	0	0	1	1	1	0
하위비트	Phase A	1	1	1	0	0	0
현재 (MCMOUT.CURHS)		5	1	3	2	6	4
다음 (MCMOUT.EXPHS)		1	3	2	6	4	5
PWM Pattern							
COUT62		0	1	1	0	0	0
CC62		0	0	0	0	1	1
COUT61		1	0	0	0	0	1
CC61		0	0	1	1	0	0
COUT60		0	0	0	1	1	0
CC60		1	1	0	0	0	0
MCMOUT.MCMPS		0x09	0x21	0x24	0x06	0x12	0x18

인버터로 들어가는 PWM 의 duty 를 변화시키면서 속도를 확인한다. 4 절에 그림 4.15 와 같이 Emo 모듈의 EmoPwm 에서 EmoPwmDutyCycle 을 50%로 설정했을 때 측정한 속도는 2000rpm으로 그림 4.31 과 같다. EmoPwmDutyCycle 을 100%로 설정하기 위해 그림 4.32 와 같이 입력하고, 모터 속도를 측정했을 때 3000rpm으로 그림 4.33 과 같다.

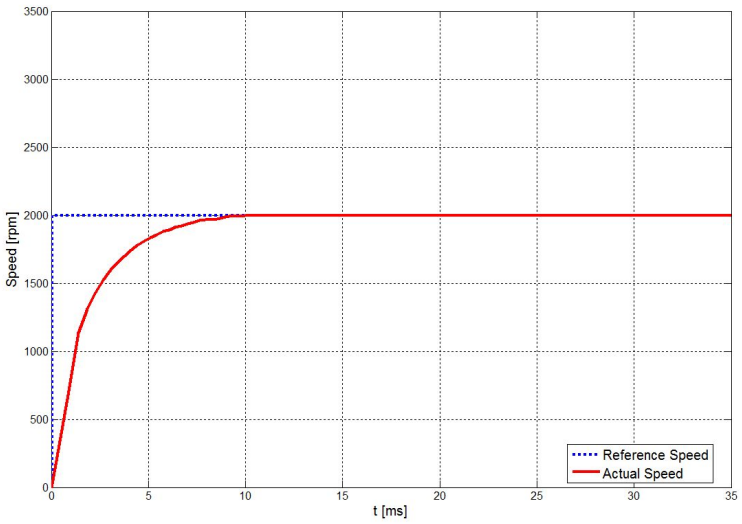


그림 4.31 BLDC 모터의 속도 측정(50% duty cycle).

Fig. 4.31 the Measurement of the speed of BLDC motor(50% duty cycle).

일 회전에 걸리는 시간당 출력되는 홀 센서의 신호로부터 속도를 계산한다. 일 회전 당 출력되는 홀 센서는 일정하지만, 일 회전 시 걸리는 시간이 모터가 구동됨에 따라 감소한다. 일 회전 시 걸리는 시간이 점차 줄어들다가 일정 값으로 고정되면, 속도는 점차 증가하다가 일정한 값을 유지한다. 또한 인버터로 들어가는 PWM 의 duty 가 증가하면 모터에 인가되는 전압이 커지게 되므로 BLDC 모터의 일 회전 당 걸리는 시간이 더 빠르게 감소하여 속도가 증가함을 확인한다.

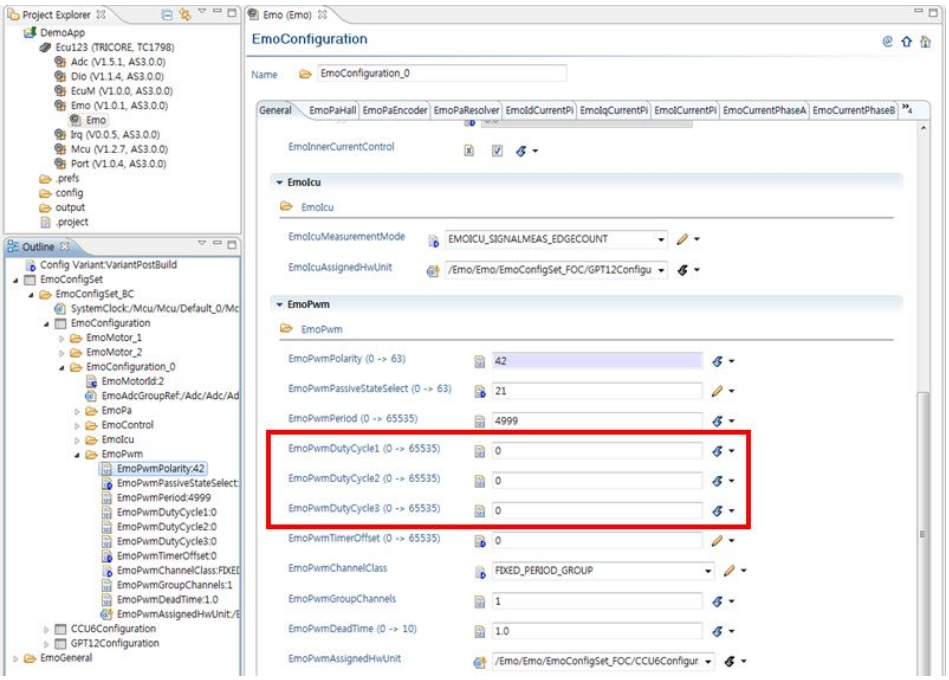


그림 4.32 Emo 모듈의 EmoPWM 변경.

Fig. 4.32 the Setting of EmoPWM for Emo Module.

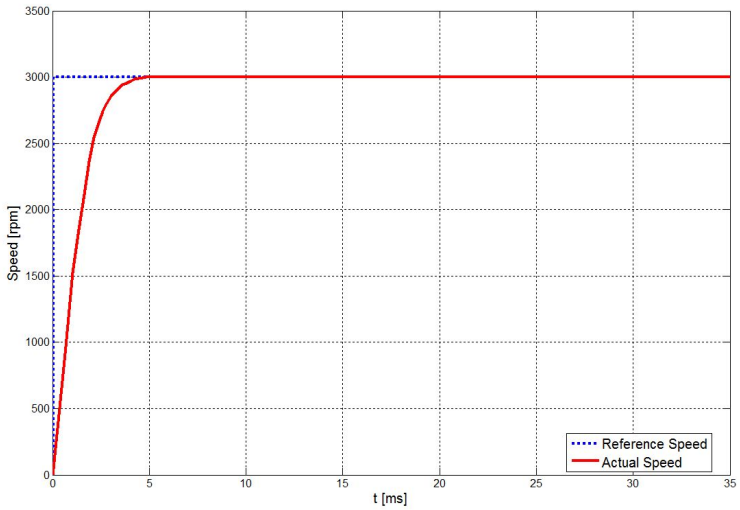


그림 4.33 BLDC 모터의 속도 측정(100% duty cycle).

Fig. 4.33 the Measurement of the speed of BLDC motor(100% duty cycle).

5. 결론

본 논문에서는 AUTOSAR 플랫폼에서 기존 안정화된 장치들과 호환성을 가지고, 직접 사용 가능한 CDD 로서 개발된 모터 전용 라이브러리인 eMotor Driver 를 사용하여 BLDC 모터 제어시스템을 구현하였다. CDD 는 AUTOSAR 에 표준화되지 않은 소프트웨어 독립체이지만, BSW 의 API 나 AUTOSAR 의 인터페이스를 통해 접근 가능하다. 또한 다른 계층들과 다르게 MCU 의 내부 인터럽트와 연관되어 있고, MCAL 계층과 통합하여 하드웨어에 직접 접근이 가능하므로 CDD 로서 개발된 eMotor Driver 와 MCAL 을 통합하여 BLDC 모터 구동에 필요한 BC Mode 를 구현하였다.

eMotor Driver 는 3 상 모터 구동에 필요한 FOC Mode, BC Mode, 전류제어기 및 서브 모듈 등이 소프트웨어적으로 분리돼있다. 따라서 하드웨어 의존성을 최소화 하고, 적은 비용으로 소프트웨어의 개발할 수 있으며 유지보수성, 재사용성, 신뢰성을 확보할 수 있다. 이와 같은 장점을 가지고 있는 eMotor Driver 를 이용하여 BLDC 모터를 구동함으로써 효율적이고 안정적인 모터 구동이 가능하다.

본 논문의 실험 시스템 구성은 Infineon 사의 32bit MCU 인 TC1798, Syncwork 사의 인버터 보드 및 BLDC 모터로 이뤄지며, BLDC 모터는 전기적 정류 방법인 BC 알고리즘을 사용하며, BC 알고리즘은 eMotor Driver 의 Hierarchy 에 정의된 소프트웨어 모듈과 MCAL 계층 모듈을 사용하여 구현하였다. 실험 결과는 BLDC 모터의 구동 특성에서 얻을 수 있는 Hall Pattern, PWM Pattern, 상(Phase) 전압을 파악하고, PWM 의 duty cycle 을 바꿀 때 모터 속도 변화에 대해 확인하였다.

참고 문헌

[1] 홍성수, 박지용, 유우선, “OSEK 과 AUTOSAR 를 중심으로 본 차량용 OS 와 미들웨어 기술 동향,” 한국정밀공학회지, vol. 23, no. 9, pp. 31-38, 2006.

[2] Klaus Grimm, “Software Technology in an Automotive Company - Major Challenges,” Procs. Of the 25th International Conference on Software Engineering, IEEE, pp. 498-509, 2003.

[3] 한무희, 이창호, “AUTOSAR 기반 BSW 에 대한 API 테스트 방안 및 사례 연구,” 한국자동차공학회 학술대회 및 전시회, pp. 1717-1720, 2011.

[4] Nico Naumann, “AUTOSAR Runtime Environment and Virtual Function Bus,” Technical report, Hasso-Plattner-Institution Softwaresystemtechlink, 2009.

[5] AUTOSAR, “Complex Driver design and integration guideline V1.0.0 R4.1 Rev 1,” Technical report, 2013.

[6] Patrick Markl, “AUTOSAR 베이직 소프트웨어를 통한 CAN-MOST 게이트웨이 구현,” AUTOMOTIVE TECHONOLOGY, 2010.

[7] Infineon, “Highly Integrated and Performance Optimized 32bit Microcontrollers for Automotive and Industrial Applications,” Technical report, 2012.

[8] 한만승, 홍성렬, 조주희, 이상훈, 박성준, 김대경, “전기자동차용 0.5[kW]급 공기압축기의 브러시리스 직류전동기 개발,” 조명·전기설비학회 논문지, 제 26 권, 제 8 호, pp. 71-78, 2012.

[9] 김상훈, “DC, AC, BLDC 모터 제어,” 북두출판사, 2010.

[10] Microsemi, “Speed Control of Brushless DC Motors-Block Commutation with Hall Sensors,” Technical report, 2012.

[11] 이강석, 박인석, 선우명호, 이우택, “차량용 전자제어시스템을 위한 AUTOSAR 대응 경량화 소프트웨어 아키텍처 연구,” 한국자동차공학회 논문집, vol. 21, no. 1, pp. 68-77, 2013.

[12] Hyun Ghul Jo, Shiquan Piao and Woo Young Jung, “Design of a Vehicular code generator for Distributed Automotive Systems,” Procs. Of the 7th International Conference on Information Technology: New Generations, pp. 1150-1153, 2010.

[13] FUJITSU Technical Analysis, “Standard Automotive Software Platform “AUTOSAR” and Microcontroller Driver MCAL Conforming to AUTOSAR Release 2.1,” vol. 25, no. 3, pp. 1-3, 2008.

[14] Wang Dafang, Zheng Jiuyang, Zhao Guifan, Huang Bo and Liu Shiqiang, “Survey of the AUTOSAR Complex Drivers in the Field of Automotive Electronics,” Procs. of the 2010 International Conference on Intelligent Computation Technology and Automation, pp. 662-664, 2010.

[15] 박광민, 금대현, 손병점, 이성훈, “AUTOSAR 기반 EPS 시스템 소프트웨어 컴포넌트의 스케줄링 설계 및 시뮬레이션,” 제어로봇시스템학회 논문지, vol. 16, no. 6, pp. 539-545, 2010.

[16] Infineon, “MC-ISAR AUDO UM EmoDriver Documentation,” Technical report, 2011.

[17] Infineon, “MC-ISAR AUDO eMotor Installation Guide,” Technical report, 2011.

[18] SyncWorks, “SMC150 개발보드 하드웨어 매뉴얼,” Technical report, 2012.

[19] Infineon, “TC1798 User’s Manual V1.1 2011-03,” Technical report, 2011.

[20] OSEK group, “OSEK/VDX Operating System Specification 2.2.3,” Technical report, 2005.

Abstract

A Study on BLDC Motor Control for Electric Vehicle using Motor Library

by Sunny Ro

Dept. of Electronics Engineering
Graduate School, Kookmin University
Seoul, Korea

This paper shows how the CDD (Complex Device Drivers) layer of the AUTOSAR (Automotive Open System Architecture) can be applied to a BLDC (Brushless DC) motor system. The eMotor Driver based CDD and an exclusive motor library provides diverse motor libraries for three-phase AC motors and allows a direct access to the hardware.

The experimental set-up consists of 32-bit microcontroller boards using eMotor Driver, BLDC motor and inverter board. It is shown by experimental results that the hall and PWM (Pulse Width Modulation) pattern of BLDC motor is used to confirm the driving BLDC motor. Also the speed response of BLDC motor is analyzed according to change PWM duty cycle.