



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2014-0023673
(43) 공개일자 2014년02월27일

(51) 국제특허분류(Int. Cl.)
G05B 17/00 (2006.01) G09B 9/00 (2006.01)
(21) 출원번호 10-2012-0089900
(22) 출원일자 2012년08월17일
심사청구일자 2012년08월17일

(71) 출원인
인하대학교 산학협력단
인천광역시 남구 인화로 100, 인하대학교 (용현동)
(72) 발명자
이영삼
인천 연수구 용담로 14, 107동 1503호 (청학동, 하나아파트)
양지혁
경기 김포시 고촌읍 수기로 8, 106동 302호 (오룡마을한화아파트)
(74) 대리인
이원희

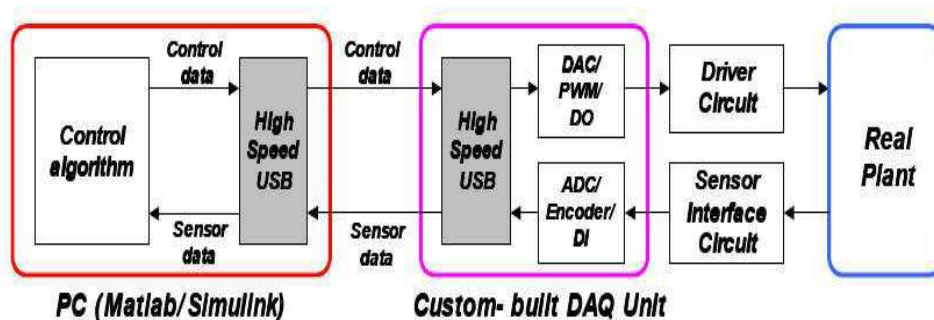
전체 청구항 수 : 총 20 항

(54) 발명의 명칭 매트랩 / 시뮬링크 및 USB DAQ 장치를 이용한 고속제어 프로토타이핑 시스템

(57) 요약

본 발명은 USB 고속(High Speed) 통신기능을 구비한 DAQ(Data Acquisition) 장치와 매트랩(Matlab)/시뮬링크(Simulink)를 기반으로 한 RCP(Rapid Controller Prototyping) 시스템에 관한 것으로, 본 발명에 따르면, 휴대성 및 비용면에 있어서 학부과정이나 대학원 학생들이 사용하기에 어려움이 많았던 기존의 RCP시스템이 가지는 문제점들을 해결하여, USB 통신을 통해 인터페이스 되는 DAQ 장치를 사용하고 C-code 생성과정 없이 바로 PC측의 Simulink가 제어가 되는 형태의 RCP 시스템이 제공됨으로써, 학부과정이나 대학원 학생들이 학교뿐만 아니라 집에서도 손쉽게 설계 및 실험을 해볼 수 있도록 하는 대중화된 RCP 시스템을 제공할 수 있다.

대표도 - 도1



(72) 발명자

김석윤

인천광역시 남구 주안 1,4동 950-25번지 금성빌라
503호

권오규

서울 양천구 목동동로 350, 536동 402호 (목동, 목
동아파트)

이 발명을 지원한 국가연구개발사업

과제고유번호 NIPA-2012-H0401-12-1007

부처명 지식경제부

연구사업명 IT융합 고급인력과정 지원사업

연구과제명 생체신호를 이용한 IT 기반 재활의료기기 개발

기 여 율 1/1

주관기관 한국산업기술대학교 산학협력단

연구기간 2011.06.01 ~ 2014.12.31

특허청구의 범위

청구항 1

고속제어 프로토타이핑(Rapid Controller Prototyping) 시스템에 있어서,

외부 센서로부터 측정된 센서 데이터를 수신하는 데이터 취득부;

상기 데이터 취득부로부터 전송되는 상기 센서 데이터에 근거하여 제어 연산을 수행하고, 처리 결과에 따른 제어 데이터를 생성하여 상기 데이터 취득부로 전송하는 데이터 처리부; 및

상기 데이터 취득부로부터 전송된 상기 제어 데이터를 표시하는 표시부를 포함하여 구성된 것을 특징으로 하는 고속제어 프로토타이핑 시스템.

청구항 2

제 1항에 있어서,

상기 데이터 취득부와 상기 데이터 처리부는 USB 고속(High Speed) 통신방식으로 서로 데이터를 전송하도록 구성된 것을 특징으로 하는 고속제어 프로토타이핑 시스템.

청구항 3

제 1항에 있어서,

상기 데이터 처리부는, 매트랩(Matlab) 및 시뮬링크(Simulink)가 실행되는 PC로 구성되는 것을 특징으로 하는 고속제어 프로토타이핑 시스템.

청구항 4

제 1항에 있어서,

상기 데이터 취득부는,

상기 센서 데이터를 수신하고 상기 제어 데이터를 상기 표시부로 전송하기 위한 입출력부; 및

상기 입출력부를 제어하기 위한 마이크로컨트롤러를 포함하여 구성되는 것을 특징으로 하는 고속제어 프로토타이핑 시스템.

청구항 5

제 4항에 있어서,

상기 입출력부는,

복수의 PWM 채널을 가지는 PWM부;

복수의 채널을 지원하는 DAC부;

복수의 채널을 지원하는 ADC부;

디지털 입력(digital input)을 지원하는 DI부;

디지털 출력(digital output)을 지원하는 DO부;

On, Off 방식으로 정보를 출력하는 LED부;

스위치(switch) 정보를 입력받을 수 있는 스위치 입력부(Switch input); 및

중분 인코더(incremental encoder)를 처리할 수 있는 인코더부(Encoder)를 포함하여 구성되는 것을 특징으로 하는 고속제어 프로토타이핑 시스템.

청구항 6

제 3항에 있어서,

상기 데이터 처리부는, 상기 시뮬링크(Simulink)의 통신 라이브러리 블록을 이용하여 상기 데이터 취득부와 통신을 행하는 것을 특징으로 하는 고속제어 프로토타이핑 시스템.

청구항 7

제 6항에 있어서,

상기 통신 라이브러리 블록은,

ENC, ADC, DI, Switch, Time 블록을 포함하는 수신(Receive) 블록; 및

PWM, DAC, DO, Buzzer, LED 블록을 포함하는 송신(Send) 블록을 포함하여 구성되는 것을 특징으로 하는 고속제어 프로토타이핑 시스템.

청구항 8

제 7항에 있어서,

상기 시뮬링크(Simulink)의 스코프(Scope) 블록을 이용하여, 전송되는 데이터를 실시간으로 모니터링 가능하도록 구성되고,

상기 매트랩(Matlab)의 워크스페이스(workspace)에 데이터를 저장 가능하도록 구성된 것을 특징으로 하는 고속제어 프로토타이핑 시스템.

청구항 9

제 8항에 있어서,

상기 시뮬링크(Simulink)의 파라미터 설정을 통하여 샘플링 주기를 설정 가능하도록 구성된 것을 특징으로 하는 고속제어 프로토타이핑 시스템.

청구항 10

제 9항에 있어서,

상기 데이터 처리부는,

상기 데이터 처리부로부터 상기 데이터 취득부로 전송해야 할 데이터의 종류를 순서대로 가지고 있는 테이블인 TxDataInfo table 및 상기 데이터 취득부로부터 상기 데이터 처리부로 전송해야 할 데이터의 종류를 순서대로 가지고 있는 테이블인 RxDataInfo table을 상기 데이터 취득부에 전송하여 통신 대상이 되는 데이터에 대한 정보를 시뮬레이션 루프(Simulation loop)가 실행되기 전에 상기 데이터 취득부에 통지하고,

상기 데이터 취득부는, 시뮬레이션 루프가 시작되면, 매번 타이머 인터럽트 서비스(timer interrupt service)가 시행될 때마다 상기 RxDataInfo table에 있는 정보를 이용하여 상기 센서 데이터를 일괄적으로 상기 데이터 처리부 측으로 전송하고, 상기 데이터 처리부가 제어 연산을 수행하고난 결과를 일괄적으로 전송하면, 상기

TxDataInfo table에 있는 정보를 이용하여 상기 제어 데이터를 해당하는 출력장치에 출력하도록 구성됨으로써, 복수의 통신 블록에 대하여도 일괄 송신 및 일괄 수신이 가능하도록 구성된 것을 특징으로 하는 고속제어 프로토타입 시스템.

청구항 11

제 10항에 있어서,

각각의 상기 통신 라이브러리 블록은, 상기 데이터 처리부에 의해 호출되는 콜백 함수(callback function)인 mdlInitializeSizes, mdlInitializeSampleTimes, mdlStart, mdlOutputs, mdlTerminate를 이용하여 구현되는 것을 특징으로 하는 고속제어 프로토타입 시스템.

청구항 12

제 11항에 있어서,

상기 mdlInitializeSizes 함수는,

mdlCheckParameters을 호출하여 각각의 상기 통신 라이브러리 블록이 전달받은 각각의 파라미터의 적합성을 검사하고, 부적합한 경우는 오류를 발생시키며,

각각의 상기 통신 라이브러리 블록이 전달받은 파라미터인 ReceiveDataInfo 또는 SendDataInfo로부터, 수신블록(Receive block)인 경우는 몇 개의 출력 포트를 가져야 하는지, 또는, 송신블록(Send block)인 경우는 몇 개의 입력 포트를 가져야 하는지를 설정하고,

처리에 사용되는 전역변수를 0으로 초기화하며, 초기화하는 작업이 필요 없을 경우는 처음 실행되었을 때에만 변수값이 0이고 두 번째 이후의 실행에는 이전 실행에서 남아 있던 값을 사용하도록 하는 처리를 수행하도록 구성된 것을 특징으로 하는 고속제어 프로토타입 시스템.

청구항 13

제 11항에 있어서,

상기 mdlInitializeSampleTimes 함수는,

각각의 상기 통신 라이브러리 블록의 샘플 시간(sample time) 정보를 설정하는 처리를 수행하도록 구성된 것을 특징으로 하는 고속제어 프로토타입 시스템.

청구항 14

제 13항에 있어서,

각각의 상기 통신 라이브러리 블록이 모두 같은 샘플 시간을 사용하도록 설정되는 것을 특징으로 하는 고속제어 프로토타입 시스템.

청구항 15

제 11항에 있어서,

상기 mdlStart 함수는,

각각의 상기 통신 라이브러리 블록에 대하여, 각각의 데이터 프레임 상에서 특정 통신 라이브러리 블록과 관련된 데이터의 순서에 해당하는 실행순서(execution order)를 나타내는 변수인 ExecOrder, 수신 데이터 프레임 상에서 특정 통신 라이브러리 블록과 관련된 데이터가 시작되는 지점을 나타내는 수신 데이터 오프셋(receive

data offset)을 나타내는 변수인 RxDataOffset 및 송신 데이터 프레임 상에서 특정 통신 라이브러리 블록과 관련된 데이터가 시작되는 지점을 나타내는 송신 데이터 오프셋(send data offset)을 나타내는 변수인 TxDataOffset을 포함하는 멤버 데이터(member data) 중 각각의 상기 통신 라이브러리 블록이 가지고 있는 멤버 데이터를 알아내어 각각 갱신하며,

갱신된 상기 멤버 데이터에 근거하여, 상기 데이터 처리부가 상기 데이터 취득부로부터 수신해야 할 데이터의 종류와 순서 정보를 가지고 있는 배열(array)인 RxDataInfo 및 상기 데이터 처리부가 상기 데이터 취득부로 전송해야 하는 데이터의 종류와 순서를 가지고 있는 배열인 TxDataInfo를 각 통신 라이브러리 블록의 실행 순서에 따라 채우고,

완성된 상기 RxDataInfo 및 상기 TxDataInfo를 시뮬레이션 루프(Simulation loop)가 시작되기 전에 상기 데이터 취득부로 송신함으로써, 상기 데이터 취득부가 상기 데이터 처리부로 송신해주어야 하는 데이터의 종류와 순서를 지정하고, 상기 데이터 취득부에 상기 데이터 처리부로부터 어떤 종류의 신호가 어떤 순서로 전송되어 올 것인지를 알려주는 처리를 수행하도록 구성된 것을 특징으로 하는 고속제어 프로토타이핑 시스템.

청구항 16

제 15항에 있어서,

상기 mdlStart 함수는,

특정 통신 라이브러리 블록이 하나의 모델에서 2번 이상 사용되었는지를 검사하고,

검사 결과 2번 이상 사용된 경우는 오류메시지를 출력하도록 하는 처리를 더 수행하도록 구성된 것을 특징으로 하는 고속제어 프로토타이핑 시스템.

청구항 17

제 11항에 있어서,

상기 mdlOutputs 함수는,

t = 0인 시점이고 실행 순서가 가장 앞서는 통신 라이브러리 블록인 경우, 함수 InitCommunication()을 호출하여 USB 통신을 위한 통신객체를 생성하고, 상기 데이터 취득부와 USB 통신을 연결하여, 상기 데이터 취득부에서 필요한 정보(SampleTimeInfo, TxDataInfo, RxDataInfo)를 전달하며,

실행 순서가 가장 앞서는 RCP 블록일 경우, 상기 데이터 취득부로부터 일괄적으로 센서 데이터를 수신하여 변수 RxBuffer에 채워 넣고, 이때, 일괄 수신된 데이터의 바이트(byte) 수가 실제 받기로 되어 있는 데이터의 바이트(byte) 수와 같은지를 확인하여, 다르다면 오류메시지 발생과 함께 처리를 종료하며,

통신 라이브러리 블록의 타입(type)이 수신 블록(Receive block)인 경우, 상기 RxBuffer로부터 해당 블록에서 필요한 데이터만을 추출하여 블록의 출력을 갱신하며,

통신 라이브러리 블록의 타입이 송신 블록(Send block)인 경우, 오프셋 정보와 채널 수 정보를 이용하여 변수 TxBuffer를 채워 넣고, 모든 송신 블록에 대한 TxBuffer에 데이터가 채워지게 되면, 상기 데이터 취득부로 일괄 전송하며,

실행 순서가 가장 늦은 통신 라이브러리 블록인 경우, 상기 TxBuffer에 채워진 데이터를 상기 데이터 취득부로 일괄 송신하는 처리를 수행하도록 구성된 것을 특징으로 하는 고속제어 프로토타이핑 시스템.

청구항 18

제 17항에 있어서,

상기 mdlOutputs 함수의 2개 이상의 조건이 동시에 만족되면, 해당하는 동작이 모두 수행되도록 구성된 것을 특징으로 하는 고속제어 프로토타이핑 시스템.

청구항 19

제 11항에 있어서,

상기 mdlTerminate 함수는,

실행 순서가 가장 늦은 통신 라이브러리 블록에서 'z'를 상기 데이터 취득부로 송신하고,

상기 데이터 취득부는, 'z'를 수신할 경우, 현재 세션(session)을 종료하고 다른 세션이 시작되는 것을 대비하여 대기 상태로 전환되며,

USB 통신을 종료하기 전에 수신 종료점(Endpoint)에 남아 있던 데이터를 읽어들이고 송신 및 수신 종료점을 리셋(reset) 시키며,

USB 통신을 위해 생성했던 객체(object)를 소멸시키는 처리를 수행하도록 구성된 것을 특징으로 하는 고속제어 프로토타이핑 시스템.

청구항 20

청구항 1항 내지 19항 중 어느 한 항에 기재된 고속제어 프로토타이핑(Rapid Controller Prototyping) 시스템을 이용하여 제어 연산을 수행하는 데이터 처리방법에 있어서,

데이터 취득부에서 제어에 필요한 센서 데이터를 측정한 후 USB 통신을 통해 데이터 처리부 측으로 전송하는 단계;

상기 데이터 취득부로부터 전송된 센서 데이터를 기반으로 상기 데이터 처리부에서 제어연산을 수행하는 단계;

상기 제어연산을 수행하는 단계에서 얻어진 제어연산 결과를 상기 데이터 취득부로 전송하는 단계; 및

상기 데이터 처리부로부터 전송된 제어데이터를 출력장치를 통해 출력하는 단계를 포함하여 구성된 것을 특징으로 하는 데이터 처리방법.

명세서

기술 분야

[0001] 본 발명은 USB 고속(High Speed) 통신기능을 구비한 DAQ(Data Acquisition) 장치와 매트랩/시뮬링크(Matlab/Simulink)를 기반으로 한 고속제어 프로토타이핑(Rapid Controller Prototyping) 시스템에 관한 것이다.

배경 기술

[0002] 일반적으로, RCP(Rapid Controller Prototyping) 시스템(이하, 'RCP 시스템'이라 함)이란, 제어시스템 개발자가 단기간에 제어 알고리즘을 효율적으로 설계, 개발, 검증하기 위해 사용하는 개발환경을 의미한다.

[0003] 또한, 전통적인 종래의 RCP 시스템은, 시뮬링크(Simulink)(이하, 'Simulink'라 함)와 같은 블록 다이어그램 방식의 모델링 프로그램, 하드웨어 입/출력장치를 제어하기 위해 사용하는 라이브러리 블록, C-code 생성기, 실시간 타겟(target) 컴퓨터, 타겟 컴퓨터와 통신하는 호스트(host) 컴퓨터 등으로 구성된다.

[0004] 즉, 기존의 RCP 시스템에서는, Simulink를 이용하여 제어를 설계하고 모의실험 등을 통해 테스트해 본 후, 만족스러운 결과가 나올 경우 입/출력 제어 라이브러리 블록을 사용하여 실제 시스템을 제어할 수 있도록 제어기

를 구성한다.

- [0005] 또한, 자동 C-code 생성기를 이용하여 구성된 제어기 모델에 대한 C-code를 생성하고 컴파일한 후, 실시간 타겟 컴퓨터에 다운로드 하여 실시간 제어 실험을 수행하게 되며, 제어 실험이 수행되는 도중 타겟 컴퓨터와 연결된 호스트 컴퓨터를 이용하여 관찰하고자 하는 신호를 실시간으로 확인해 보고, 특정 파라미터를 실시간으로 튜닝할 수도 있다.
- [0006] 아울러, RCP 시스템은 하드웨어 입/출력장치를 제어하기 위한 Simulink용 라이브러리 블록이 제공되고, C-code 생성 기능이 포함되어 있으므로, 제어 시스템 개발자는 지루하고 오류가 발생하기 쉬운 수작업 코딩(coding)을 하지 않아도 되고, 하드웨어 장치를 제어하는 것에 대한 부담 없이 오로지 제어 알고리즘의 개발이라는 본연의 목적에만 집중할 수 있다.
- [0007] 이러한 RCP 시스템은, 특히, ECU(Engine Control Unit) 개발과 같이, 자동차산업에서 활발하게 사용되고 있으며, RCP 시스템과 관련된 몇몇 소프트웨어가 상업적으로 판매되고 있는데, 그 중 매트랩/시뮬링크(Matlab/Simulink)는 가장 잘 알려지고 널리 사용되는 프로그램이다.
- [0008] 또한, Simulink의 추가 제품인 리얼타임 워크샵(Real-time Workshop, RTW)은, Simulink로 구성된 블록 다이어그램 모델에 대해 자동으로 ANSI-C code를 생성해 준다(참고문헌 1 참조).
- [0009] 아울러, Simulink의 또 다른 추가제품인 임베디드 코더(Embedded Coder)는 특정 내장형 프로세서에 특화된 C-code를 생성해 줌으로써 제품 생산기간을 단축해 주는 기능을 제공해 준다(참고문헌 2 참조).
- [0010] 더욱이, Matlab/Simulink와 RTW는, 개방형 구조를 취함으로써 직접 개발한 하드웨어 장치에 대해서도 RCP 소프트웨어를 적용할 수 있도록 하고 있으며, 이러한 개방성에 의해, 자체 개발된 하드웨어에 Matlab/Simulink/RTW를 사용한 RCP 시스템들이 학술적인 연구로써 다수 제안되었다(참고문헌 3 내지 7 참조).
- [0011] 또한, 최근, 국내의 상당수 공과대학에서는 공학교육의 내실을 기하기 위해 공학교육인증 교과과정을 채택하고 있으며, 공학교육인증 교과과정의 특징 중 하나는 실험 수업과 설계 수업에 큰 비중을 두고 있다는 것이나, 일반적으로 공과대학에서 이루어지는 학생들의 실험/실습 수업은 학교의 실험실 내에서만 이루어지고 학생들의 집에서 개인적인 실험/실습은 이루어지고 있지 않다.
- [0012] 이는, 수업에 사용되는 장비와 소프트웨어가 학교에만 설치되어 있기 때문으로, 만일 학생들이 학교의 실험/실습실 이외에 집에서도 실험 및 설계를 수행해 볼 수 있다면 좀 더 빠르게 수업에 익숙해질 수 있게 되어 수업효율을 크게 향상시킬 수 있을 것이다.
- [0013] 아울러, RCP 시스템을 활용한 제어 시스템 설계 및 실험은 학생들에게 수작업 코딩의 번거로움을 덜게 하고, 제어 알고리즘 설계라는 본연의 목적에 충실할 수 있게 함으로써, 과목에 대한 이해와 설계 및 실험에 대한 학생들의 흥미를 크게 증진할 수 있을 것이다.
- [0014] 그러나 상업적으로 판매되는 RCP 시스템이나 학술적인 연구로 개발된 RCP 시스템은 모두 고가의 하드웨어 장치와 소프트웨어를 필요로 하거나, 휴대성이 떨어지는 하드웨어 장치를 기반으로 하기 때문에, 학생들이 집에서도 설계 및 실험을 해볼 수 있게 하기에는 현실적으로 많은 어려움이 따른다.

- [0015] 예를 들면, PC를 타겟 컴퓨터로 사용하고 DAQ(Data acquisition) 장치를 사용하여 입출력을 처리하는 RCP 시스템의 경우, DAQ 장치가 ISA 또는 PCI 버스 방식으로 인터페이스 되기 때문에 특정 PC에 장착되어 있어야 한다.
- [0016] 또한, 휴대가 가능한 임베디드 보드를 타겟으로 사용하는 RCP 시스템의 경우, 임베디드 코더(Embedded coder)를 사용하고 특정 프로세서용 컴파일러를 필요로 한다.
- [0017] 그러나 이들은 모두 고가의 소프트웨어이기 때문에 학생들이 개별적으로 소유할 수는 없다.
- [0018] 따라서 상기한 바와 같이, 휴대성 및 비용면에 있어서 학부과정이나 대학원 학생들이 사용하기에 어려움이 많았던 기존의 RCP시스템이 가지는 문제점들을 해결하여, 학부과정이나 대학원 학생들이 학교뿐만 아니라 집에서도 손쉽게 설계 및 실험을 해볼 수 있도록 하는 새로운 RCP 시스템을 제공하는 것이 바람직하나, 아직까지 그러한 요구를 모두 만족시키는 장치나 방법은 제공되지 못하고 있는 실정이다.

발명의 내용

해결하려는 과제

- [0019] 본 발명은 상기한 바와 같은 종래의 RCP 시스템의 문제점을 해결하고자 하는 것으로, 따라서 본 발명의 목적은, 상기한 바와 같은 기존의 RCP시스템이 가지는 휴대성 및 비용면에 있어서의 문제점들을 해결하고, 학부과정이나 대학원 학생들이 학교뿐만 아니라 집에서도 손쉽게 설계 및 실험을 해볼 수 있도록 하는 대중화된 RCP 시스템을 제공하고자 하는 것이다.

과제의 해결 수단

- [0020] 상기한 바와 같은 목적을 달성하기 위해, 본 발명에 따르면, 고속제어 프로토타이핑(Rapid Controller Prototyping) 시스템에 있어서, 외부 센서로부터 측정된 센서 데이터를 수신하는 데이터 취득부; 상기 데이터 취득부로부터 전송되는 상기 센서 데이터에 근거하여 제어 연산을 수행하고, 처리 결과에 따른 제어 데이터를 생성하여 상기 데이터 취득부로 전송하는 데이터 처리부; 및 상기 데이터 취득부로부터 전송된 상기 제어 데이터를 표시하는 표시부를 포함하여 구성된 것을 특징으로 하는 고속제어 프로토타이핑 시스템이 제공된다.
- [0021] 여기서, 상기 데이터 취득부와 상기 데이터 처리부는 USB 고속(High Speed) 통신방식으로 서로 데이터를 전송하도록 구성된 것을 특징으로 한다.
- [0022] 또한, 상기 데이터 처리부는, 매트랩(Matlab) 및 시뮬링크(Simulink)가 실행되는 PC로 구성되는 것을 특징으로 한다.
- [0023] 아울러, 상기 데이터 취득부는, 상기 센서 데이터를 수신하고 상기 제어 데이터를 상기 표시부로 전송하기 위한 입출력부; 및 상기 입출력부를 제어하기 위한 마이크로컨트롤러를 포함하여 구성되는 것을 특징으로 한다.
- [0024] 더욱이, 상기 입출력부는, 복수의 PWM 채널을 가지는 PWM부; 복수의 채널을 지원하는 DAC부; 복수의 채널을 지원하는 ADC부; 디지털 입력(digital input)을 지원하는 DI부; 디지털 출력(digital output)을 지원하는 DO부; On, Off 방식으로 정보를 출력하는 LED부; 스위치(switch) 정보를 입력받을 수 있는 스위치 입력부(Switch

input); 및 증분 인코더(incremental encoder)를 처리할 수 있는 인코더부(Encoder)를 포함하여 구성되는 것을 특징으로 한다.

[0025] 또한, 상기 데이터 처리부는, 상기 시뮬링크(Simulink)의 통신 라이브러리 블록을 이용하여 상기 데이터 취득부와 통신을 행하는 것을 특징으로 한다.

[0026] 아울러, 상기 통신 라이브러리 블록은, ENC, ADC, DI, Switch, Time 블록을 포함하는 수신(Receive) 블록; 및 PWM, DAC, DO, Buzzer, LED 블록을 포함하는 송신(Send) 블록을 포함하여 구성되는 것을 특징으로 한다.

[0027] 더욱이, 상기한 고속제어 프로토타이핑 시스템은, 상기 시뮬링크(Simulink)의 스코프(Scope) 블록을 이용하여, 전송되는 데이터를 실시간으로 모니터링 가능하도록 구성되고, 상기 매트랩(Matlab)의 워크스페이스(workspace)에 데이터를 저장 가능하도록 구성된 것을 특징으로 한다.

[0028] 또한, 상기한 고속제어 프로토타이핑 시스템은, 상기 시뮬링크(Simulink)의 파라미터 설정을 통하여 샘플링 주기를 설정 가능하도록 구성된 것을 특징으로 한다.

[0029] 아울러, 상기 데이터 처리부는, 상기 데이터 처리부로부터 상기 데이터 취득부로 전송해야 할 데이터의 종류를 순서대로 가지고 있는 테이블인 TxDataInfo table 및 상기 데이터 취득부로부터 상기 데이터 처리부로 전송해야 할 데이터의 종류를 순서대로 가지고 있는 테이블인 RxDataInfo table을 상기 데이터 취득부에 전송하여 통신 대상이 되는 데이터에 대한 정보를 시뮬레이션 루프(Simulation loop)가 실행되기 전에 상기 데이터 취득부에 통지하고, 상기 데이터 취득부는, 시뮬레이션 루프가 시작되면, 매번 타이머 인터럽트 서비스(timer interrupt service)가 시행될 때마다 상기 RxDataInfo table에 있는 정보를 이용하여 상기 센서 데이터를 일괄적으로 상기 데이터 처리부 측으로 전송하고, 상기 데이터 처리부가 제어 연산을 수행하고난 결과를 일괄적으로 전송하면, 상기 TxDataInfo table에 있는 정보를 이용하여 상기 제어 데이터를 해당하는 출력장치에 출력하도록 구성됨으로써, 복수의 통신 블록에 대하여도 일괄 송신 및 일괄 수신이 가능하도록 구성된 것을 특징으로 한다.

[0030] 더욱이, 각각의 상기 통신 라이브러리 블록은, mdlCheckParameters, mdlInitializeSizes, mdlInitializeSampleTimes, mdlStart, mdlOutputs, mdlTerminate를 각각 포함하고 상기 데이터 처리부에 의해 호출되는 콜백 함수(callback function)를 이용하여 구현되는 것을 특징으로 한다.

[0031] 여기서, 상기 mdlInitializeSizes 함수는, mdlCheckParameters을 호출하여 각각의 상기 통신 라이브러리 블록이 전달받은 각각의 파라미터의 적합성을 검사하고, 부적합한 경우는 오류를 발생시키며, 각각의 상기 통신 라이브러리 블록이 전달받은 파라미터인 ReceiveDataInfo 또는 SendDataInfo로부터, 수신블록(Receive block)인 경우는 몇 개의 출력 포트를 가져야 하는지, 또는, 송신블록(Send block)인 경우는 몇 개의 입력 포트를 가져야 하는지를 설정하고, 처리에 사용되는 전역변수를 0으로 초기화하며, 초기화하는 작업이 필요 없을 경우는 처음 실행되었을 때에만 변수값이 0이고 두 번째 이후의 실행에는 이전 실행에서 남아 있던 값을 사용하도록 하는 처리를 수행하도록 구성된 것을 특징으로 한다.

[0032] 또한, 상기 mdlInitializeSampleTimes 함수는, 각각의 상기 통신 라이브러리 블록의 샘플 시간(sample time) 정보를 설정하는 처리를 수행하도록 구성된 것을 특징으로 한다.

[0033] 여기서, 상기한 시스템은, 각각의 상기 통신 라이브러리 블록이 모두 같은 샘플 시간을 사용하도록 설정하는 것을 특징으로 한다.

- [0034] 아울러, 상기 mdlStart 함수는, 각각의 상기 통신 라이브러리 블록에 대하여, 각각의 데이터 프레임 상에서 특정 통신 라이브러리 블록과 관련된 데이터의 순서에 해당하는 실행순서(execution order)를 나타내는 변수인 ExecOrder, 수신 데이터 프레임 상에서 특정 통신 라이브러리 블록과 관련된 데이터가 시작되는 지점을 나타내는 수신 데이터 오프셋(receive data offset)을 나타내는 변수인 RxDataOffset 및 송신 데이터 프레임 상에서 특정 통신 라이브러리 블록과 관련된 데이터가 시작되는 지점을 나타내는 송신 데이터 오프셋(send data offset)을 나타내는 변수인 TxDataOffset을 포함하는 멤버 데이터(member data) 중 각각의 상기 통신 라이브러리 블록이 가지고 있는 멤버 데이터를 알아내어 각각 갱신하며, 갱신된 상기 멤버 데이터에 근거하여, 상기 데이터 처리부가 상기 데이터 취득부로부터 수신해야 할 데이터의 종류와 순서 정보를 가지고 있는 배열(array)인 RxDataInfo 및 상기 데이터 처리부가 상기 데이터 취득부로 전송해야 하는 데이터의 종류와 순서를 가지고 있는 배열인 TxDataInfo를 각 통신 라이브러리 블록의 실행 순서에 따라 채우고, 완성된 상기 RxDataInfo 및 상기 TxDataInfo를 시뮬레이션 루프(Simulation loop)가 시작되기 전에 상기 데이터 취득부로 송신함으로써, 상기 데이터 취득부가 상기 데이터 처리부로 송신해주어야 하는 데이터의 종류와 순서를 지정하고, 상기 데이터 취득부에 상기 데이터 처리부로부터 어떤 종류의 신호가 어떤 순서로 전송되어 올 것인지를 알려주는 처리를 수행하도록 구성된 것을 특징으로 한다.
- [0035] 더욱이, 상기 mdlStart 함수는, 특정 통신 라이브러리 블록이 하나의 모델에서 2번 이상 사용되었는지를 검사하고, 검사 결과 2번 이상 사용된 경우는 오류메시지를 출력하도록 하는 처리를 더 수행하도록 구성된 것을 특징으로 한다.
- [0036] 또한, 상기 mdlOutputs 함수는, $t = 0$ 인 시점이고 실행 순서가 가장 앞서는 통신 라이브러리 블록인 경우, 함수 InitCommunication()을 호출하여 USB 통신을 위한 통신객체를 생성하고, 상기 데이터 취득부와 USB 통신을 연결하여, 상기 데이터 취득부에서 필요한 정보(SampleTimeInfo, TxDataInfo, RxDataInfo)를 전달하며, 실행 순서가 가장 앞서는 RCP 블록일 경우, 상기 데이터 취득부로부터 일괄적으로 센서 데이터를 수신하여 변수 RxBuffer에 채워 넣고, 이때, 일괄 수신된 데이터의 바이트(byte) 수가 실제 받기로 되어 있는 데이터의 바이트(byte) 수와 같은지를 확인하여, 다르다면 오류메시지 발생과 함께 처리를 종료하며, 통신 라이브러리 블록의 타입(type)이 수신 블록(Receive block)인 경우, 상기 RxBuffer로부터 해당 블록에서 필요한 데이터만을 추출하여 블록의 출력을 갱신하며, 통신 라이브러리 블록의 타입이 송신 블록(Send block)인 경우, 오프셋 정보와 채널 수 정보를 이용하여 변수 TxBuffer를 채워 넣고, 모든 송신 블록에 대한 TxBuffer에 데이터가 채워지게 되면, 상기 데이터 취득부로 일괄 전송하며, 실행 순서가 가장 늦은 통신 라이브러리 블록인 경우, 상기 TxBuffer에 채워진 데이터를 상기 데이터 취득부로 일괄 송신하는 처리를 수행하도록 구성된 것을 특징으로 한다.
- [0037] 여기서, 상기 mdlOutputs 함수의 2개 이상의 조건이 동시에 만족되면, 해당하는 동작이 모두 수행되도록 구성된 것을 특징으로 한다.
- [0038] 아울러, 상기 mdlTerminate 함수는, 실행 순서가 가장 늦은 통신 라이브러리 블록에서 'z'를 상기 데이터 취득부로 송신하고, 상기 데이터 취득부는, 'z'를 수신할 경우, 현재 세션(session)을 종료하고 다른 세션이 시작되는 것을 대비하여 대기 상태로 전환되며, USB 통신을 종료하기 전에 수신 종료점(Endpoint)에 남아 있던 데이터를 읽어들이고 송신 및 수신 종료점을 리셋(reset) 시키며, USB 통신을 위해 생성했던 객체(object)를 소멸시키는 처리를 수행하도록 구성된 것을 특징으로 한다.
- [0039] 더욱이, 본 발명에 따르면, 상기에 기재된 고속제어 프로토타이핑(Rapid Controller Prototyping) 시스템을 이용하여 제어 연산을 수행하는 데이터 처리방법에 있어서, 데이터 취득부에서 제어에 필요한 센서 데이터를 측정 한 후 USB 통신을 통해 데이터 처리부 측으로 전송하는 단계; 상기 데이터 취득부로부터 전송된 센서 데이터를 기반으로 상기 데이터 처리부에서 제어연산을 수행하는 단계; 상기 제어연산을 수행하는 단계에서 얻어진 제어 연산 결과를 상기 데이터 취득부로 전송하는 단계; 및 상기 데이터 처리부로부터 전송된 제어데이터를 출력장치

를 통해 출력하는 단계를 포함하여 구성된 것을 특징으로 하는 데이터 처리방법이 제공된다.

발명의 효과

- [0040] 상기한 바와 같이, 본 발명에 따르면, 기존의 RCP시스템이 가지는 휴대성 및 비용면에 있어서의 문제점들을 해결하고, 학부과정이나 대학원 학생들이 학교뿐만 아니라 집에서도 손쉽게 설계 및 실험을 해볼 수 있도록 하는 대중화된 RCP 시스템을 제공할 수 있다.
- [0041] 즉, 본 발명에 따르면, 본 발명자들에 의해 자체 개발된 RCP 시스템이 제공됨으로써, 기존의 RCP 시스템 및 대학교에서 사용하는 학부 또는 대학원용 실험실습 기자재들이 대부분 자체개발이 아니라 외부에서 구매한 것이므로 유지, 보수에 어려움을 겪었던 문제점을 해결하여, 유지, 보수의 편리성 면에서 큰 장점을 가지는 RCP 시스템을 제공할 수 있다.
- [0042] 또한, 본 발명에 따르면, 리얼타임 워크샵(Realtime Workshop)이나 임베디드 코더(Embedded coder)와 같이 기존의 RCP 시스템에서 필요로 하는 고가의 소프트웨어를 구매할 필요 없이, 많은 대학교에서 실험 실습용으로 구비하고 있는 가장 기본적인 프로그램인 매트랩(Matlab)과 시뮬링크(Simulink) 만으로 구성 가능하여 비용 면에서 매우 경제적인 RCP 시스템을 제공할 수 있다.
- [0043] 더욱이, 본 발명에 따르면, 기존의 RCP 시스템과는 달리 Simulink 자체가 제어기 기능을 수행하고 C-code를 생성하는 과정과 컴파일의 필요 없으므로, 제어를 변경하고 실험해 보는 데 소요되는 시간이 크게 단축될 수 있으며, 또한, DAQ 장치가 USB 통신을 사용하여 장치가 특정 PC에 종속될 필요가 없으므로, 학교에서만 아니라 집에서도 실험이 가능하여 학생들의 학습 효율향상에 크게 기여할 수 있는 RCP 시스템을 제공할 수 있다.

도면의 간단한 설명

- [0044] 도 1은 본 발명의 실시예에 따른 Matlab과 DAQ 장치를 이용한 RCP 시스템의 전체적인 구성을 개략적으로 나타내는 도면이다.
- 도 2는 도 1에 나타난 본 발명의 실시예에 따른 Matlab과 DAQ 장치를 이용한 RCP 시스템의 처리과정을 나타내는 흐름도이다.
- 도 3은 본 발명의 실시예에 따른 Matlab과 DAQ 장치를 이용한 RCP 시스템에 적용된 DAQ 장치의 실제 사진을 나타내는 도면이다.
- 도 4는 도 3에 나타난 DAQ 장치의 하드웨어 구성을 개념적으로 나타내는 도면이다.
- 도 5는 본 발명의 실시예에 따른 RCP 시스템의 Simulink용 통신 라이브러리 블록을 나타내는 도면이다.
- 도 6은 대화상자를 통하여 채널의 지정 및 블록의 형태를 변경하는 화면을 나타내는 도면이다.
- 도 7은 본 발명의 실시예에 따른 RCP 시스템을 이용하여 독립진자의 실시간 제어 실험을 수행하기 위해 자체 개발한 독립진자의 실물 사진을 나타내는 도면이다.
- 도 8은 본 발명의 실시예에 따른 RCP 시스템에서 제공되는 통신 라이브러리 블록과 Simulink에서 제공하는 기능 블록들을 조합하여 구성한 독립진자 시스템의 스윙업 제어기 모델의 전체적인 구성을 개략적으로 나타내는 도면이다.
- 도 9는 본 발명의 실시예에 따른 RCP 시스템을 이용하여 도 7에 나타난 독립진자의 스윙업 제어실험을 수행한 실험결과를 나타내는 도면이다.
- 도 10은 Simulink가 simulation을 수행하는 방식을 나타내는 흐름도이다.
- 도 11은 Simulink의 동작방식을 설명하기 위한 Simulink 모델의 구성예를 나타내는 도면이다.

도 12는 본 발명의 실시예에 따른 RCP 시스템에서 제공하는 10개의 통신 라이브러리 블록을 모두 사용하는 경우의 동작을 테스트하기 위해 구성한 Simulink 모델이다.

도 13은 도 12에 나타낸 Simulink 모델에 대한 RxDataInfo 및 TxDataInfo 테이블의 구성을 나타내는 도면이다.

도 14는 샘플링 주기를 8ms로 하여 도 12에 나타낸 Simulink 모델에 대한 실시간 수행 성능을 시험한 결과를 나타내는 도면이다.

도 15는 샘플링 주기를 1ms로 하여 도 12에 나타낸 Simulink 모델에 대한 실시간 수행 성능을 시험한 결과를 나타내는 도면이다.

도 16은 RCP 블록에 사용된 S-Function 블록의 대화상자를 나타내는 도면이다.

도 17은 RCP 블록의 종류와 Block Type Number를 정리하여 표로 나타낸 도면이다.

도 18은 ReceiveDataInfo와 SendDataInfo의 내용을 표로 정리하여 나타낸 도면이다.

도 19는 Simulink에서 실행 순서를 확인해 보기 위해 구성한 Simulink model을 나타내는 도면이다.

도 20은 도 19의 모델에서 Source block과 Sink block 대신 Receive Block과 Send Block을 사용해서 구성한 Simulink 모델을 나타내는 도면이다.

도 21은 Simulink와 S-function의 상호작용을 나타내는 도면으로, Simulink가 어떤 순서로 S-function의 callback method를 호출하는지를 나타내는 도면이다.

도 22는 S-function 블록이 Simulink가 관리하는 다양한 데이터와 상호작용하는 방식을 나타내는 도면이다.

도 23은 AvrDaqSfun 블록이 상호작용하는 method와 데이터를 나타내는 도면이다.

도 24는 도 2의 흐름도를 S-function의 callback method를 통해 나타내는 동시에 같이 통신하면서 수행되는 DAQ 장치의 동작 흐름도를 구체화하여 나타내는 도면이다.

도 25는 callback method 중에서 mdlInitializeSizes와 mdlStart의 처리를 나타내는 플로차트이다.

도 26은 callback method 내에서 호출되는 주요 함수인 InitCommunication()의 처리를 나타내는 플로차트이다.

도 27은 callback method인 mdlOutputs()의 처리를 나타내는 플로차트이다.

도 28은 본 발명의 실시예에 따른 RCP 시스템의 수신 데이터 프레임의 구성을 나타내는 도면이다.

도 29는 본 발명의 실시예에 따른 RCP 시스템의 수신 데이터 프레임의 구성을 나타내는 도면이다.

도 30은 도 12에 나타낸 Simulink 모델에 사용된 RCP 블록들의 멤버 데이터를 표로 정리하여 나타내는 도면이다.

도 31은 도 12의 Simulink 모델에 대해 각각의 RCP 블록에 대한 mdlStart가 모두 완료되면 생성되는 TxDataInfo와 RxDataInfo의 구조를 나타내는 도면이다.

발명을 실시하기 위한 구체적인 내용

[0045] 이하, 첨부된 도면을 참조하여 본 발명에 따른 Matlab과 DAQ 장치를 이용한 RCP 시스템의 상세한 내용에 대하여 설명한다.

[0046] 여기서, 이하에 설명하는 내용은 본 발명을 실시하기 위한 실시예일 뿐이며, 본 발명은 이하에 설명하는 실시예의 내용으로만 한정되는 것은 아니라는 사실에 유념해야 한다.

[0047] 즉, 본 발명은, 후술하는 바와 같이, 기존의 RCP시스템이 가지는 휴대성 및 비용면에 있어서의 문제점들을 해결하고, 학부과정이나 대학원 학생들이 학교뿐만 아니라 집에서도 손쉽게 설계 및 실험을 해볼 수 있도록 하는 대중화된 RCP 시스템을 제공하고자 하는 것이다.

- [0048] 여기서, 휴대성이 좋은 하드웨어 장치를 만들기 위한 하나의 방법은, 해당 장치를 USB로 인터페이스 하는 것이다.
- [0049] 더 상세하게는, 제어가 목적이 아니라 단순히 데이터 수집이 목적인 경우, 이미 USB를 이용한 DAQ 장치를 사용하는 방법들이 많이 시도되고 있다(참고문헌 8, 9, 10 참조).
- [0050] 즉, USB는 플러그 앤 플레이(plug-and-play)를 지원하고 장치를 PC에 접속할 때 PC의 전원을 끌 필요가 없기 때문에, 하드웨어 장치의 휴대성을 극대화시킬 수 있다.(참고문헌 11 참조)
- [0051] 그러나 기존의 RCP 시스템에서는 USB 통신 방식의 하드웨어 장치를 사용하는 방법을 사용하지 않고 있으며, 이는 USB 통신 방식이 실시간 제어에 필요한 소량 데이터의 고속 전송에 적합하지 않기 때문이다.
- [0052] 그러나 학부나 대학원 학생들의 설계 및 실험에는 아주 빠른 고속의 샘플링이 필요하지 않을 가능성이 높기 때문에 USB를 이용한 하드웨어 장치의 사용이 가능한 대안이 될 수 있다.
- [0053] 또한, 고속(High speed) USB는 풀스피드(Full speed) USB에 비해 상대적으로고속 통신이 가능하기 때문에 High speed 통신방식을 사용한다면 통신속도도 크게 개선할 수 있을 것이다.
- [0054] 이는, 임베디드 코더(Embedded coder)와 컴파일러 같은 고가의 소프트웨어 사용은 기본적으로 RCP 시스템이 C-code 생성을 기반으로 하고 있기 때문이다.
- [0055] 따라서 코드 생성을 사용하지 않고 Simulink 만을 사용해서 RCP시스템을 구성할 수 있다면 비용에 대한 문제도 해결할 수 있을 것이다.
- [0056] 실제로 추가 구입 제품을 포함하지 않는 기본 Matlab과 Simulink는 많은 학교에서 구비하고 있고, 이 소프트웨어들은 서버에 접속해 특정 인원 내에서 동시 사용이 가능한 형태로 사용하기 때문에 학생들이 집에서도 사용해 볼 수가 있다.
- [0057] 따라서 상기한 바와 같은 방법으로 소프트웨어에 대한 비용문제를 해결할 수 있다.
- [0058] 이에 따라, 본 발명자들은, 상기한 바와 같은 내용에 근거하여 기존의 RCP 시스템이 가지는 휴대성 문제와 비용 문제를 해결할 수 있는 새로운 RCP 시스템을 제안하였다.
- [0059] 즉, 본 발명은, USB 통신을 통해 인터페이스 되는 DAQ 장치를 사용하고, C-code 생성과정 없이 바로 PC측의 Simulink가 제어가 되는 형태의 RCP 시스템을 제안하여, 학부과정이나 대학원 학생들이 학교뿐만 아니라 집에서도 손쉽게 설계 및 실험을 해볼 수 있도록 하는 대중화된 RCP 시스템을 제공하고자 하는 것이다.
- [0060] 이하, 첨부된 도면을 참조하여, 상기한 바와 같은 본 발명에 따른 Matlab과 DAQ 장치를 이용한 RCP 시스템의 구체적인 실시예에 대하여 설명한다.

- [0061] 먼저, 도 1을 참조하면, 도 1은 본 발명의 실시예에 따른 Matlab과 DAQ 장치를 이용한 RCP 시스템의 전체적인 구성을 개략적으로 나타내는 도면이다.
- [0062] 도 1에 나타난 바와 같이, 본 발명의 실시예에 따른 Matlab과 DAQ 장치를 이용한 RCP 시스템은, 플랜트를 제외하고 크게 2개의 하부 시스템(subsystem)으로 구성된다.
- [0063] 첫 번째 하부 시스템은 MS Windows 상에서 Matlab/Simulink가 실행되는 PC 시스템이고, 두 번째 하부 시스템은 본 발명자들에 의해 자체 개발된 DAQ 장치로 구성된다.
- [0064] 여기서, PC와 DAQ 장치는 고속(High speed) USB 통신을 이용하여 데이터(제어 데이터 및 센서 데이터)를 주고 받는다.
- [0065] 또한, 본 발명의 실시예에 따른 RCP 시스템은, Simulink를 이용하여 구성된 제어기 모델에 대해 C-code를 자동 생성하고 컴파일하여 내장형 제어장치(Embedded control unit)에 다운로드 하여 제어기를 실행시키는 기존의 RCP 시스템과 달리, code 생성과정 없이 바로 Simulink를 실행함으로써 제어기를 실행하는 점이 특징이다.
- [0066] 아울러, 본 발명의 실시예에 따른 RCP 시스템에서 제어가 수행되는 방식을 정리하면 다음과 같다.
- [0067] 1. DAQ 장치가 제어에 필요한 센서 데이터를 측정 후 USB 통신을 통해 PC측으로 전송한다.
- [0068] 2. PC에서 실행중인 Simulink는 DAQ 장치에서 전송된 센서 데이터를 기반으로 제어연산을 수행한다.
- [0069] 3. 계산된 제어연산 결과를 DAQ 장치로 전송한다.
- [0070] 4. DAQ 장치에서는 PC측에서 전송된 제어데이터를 출력장치(DAC, PWM 등등)를 통해 출력한다.
- [0071] 다음으로, 도 2를 참조하면, 도 2는 도 1에 나타난 본 발명의 실시예에 따른 Matlab과 DAQ 장치를 이용한 RCP 시스템의 처리과정을 나타내는 흐름도이다.
- [0072] 도 2에 있어서, DAQ 장치가 PC 측으로 송신해야 할 센서 데이터와 PC로부터 수신받을 제어 데이터의 종류는 Simulink 상에서 수행되는 제어기의 구성에 따라 달라지게 된다.
- [0073] 또한, 제어기는, Simulink에서 지원하는 여러 가지 기능 블록과 본 RCP 시스템을 위해 개발된 입/출력 처리용 통신 블록을 이용하여 구성된다.
- [0074] 아울러, 본 발명의 실시예에 따른 RCP 시스템에서, 제어 연산은 전적으로 PC에서 수행되며, DAQ 장치는 센서 데이터의 측정과 구동신호를 출력하는 기능을 수행한다.
- [0075] 이러한 제어 연산을 일정한 주기로 실시간 수행하기 위해, DAQ 장치에 탑재된 마이크로 컨트롤러

(microcontroller)의 타이머 인터럽트(timer interrupt) 기능을 활용한다.

- [0076] 즉, 상기한 제어가 수행되는 방식에서 1번과 4번은 DAQ 장치 쪽에서 실행되고, 2번과 3번은 PC 쪽의 Simulink에서 실행된다.
- [0077] 또한, 1번과 4번에 해당하는 부분을 타이머 인터럽트 서비스 루틴(timer interrupt service routine)으로 처리하면, 비록 2번과 3번이 PC 쪽에서 실행되어 전체 과정이 약간의 편차를 보이긴 하지만 일정한 주기로 실행되게 된다.
- [0078] 계속해서, 도 3 및 도 4를 참조하여, 본 발명의 실시예에 따른 Matlab과 DAQ 장치를 이용한 RCP 시스템에 적용된 DAQ 장치의 구체적인 내용에 대하여 설명한다.
- [0079] 먼저, 도 3을 참조하면, 도 3은 본 발명의 실시예에 따른 Matlab과 DAQ 장치를 이용한 RCP 시스템에 적용된 DAQ 장치의 실제 사진을 나타내는 도면이다.
- [0080] 또한, 도 4를 참조하면, 도 4는 도 3에 나타난 DAQ 장치의 하드웨어 구성을 개념적으로 나타내는 도면이다.
- [0081] 즉, 도 3 및 도 4에 나타난 바와 같이, 본 발명의 실시예에 따른 RCP 시스템에서 사용되는 DAQ 장치는, 다양한 실험을 수행할 수 있도록 여러 종류의 입/출력장치들을 탑재하고, High speed USB 통신 기능을 지원하도록 자체 개발된 것으로, 상기한 바와 같이 제어 연산에는 관여하지 않고, 제어 연산에 필요한 센서 데이터를 측정하여 PC로 전송하고 PC측의 Simulink가 계산한 제어 데이터를 수신받아 출력장치를 통해 출력하는 기능을 수행한다.
- [0082] 또한, 도 3 및 도 4에 나타난 DAQ 장치에 탑재된 입/출력 장치와 통신 장치를 제어하기 위한 마이크로컨트롤러(microcontroller)로는 ATMEGA128을 사용하였다.
- [0083] 여기서, ATMEGA128은 16MHz의 클럭을 사용하는 8-bit 저가형 마이크로컨트롤러로서, GCC 컴파일러라는 무료 컴파일러가 지원되며, 다양한 응용사례에 대한 자료들을 손쉽게 찾을 수 있기 때문에 개발에 용이한 장점이 있다.
- [0084] 즉, 본 발명의 실시예에 따른 RCP 시스템에서는 고속의 계산이 필요한 제어연산은 Simulink가 수행되는 PC 쪽에서 수행되므로, DAQ 장치에 사용되는 컨트롤러는 고속연산을 지원하지 않아도 시스템의 성능에 큰 영향을 미치지 않는다.
- [0085] 또한, 상기한 DAQ 장치는 High speed USB 통신기능을 지원하고, USB는 플러그 앤 플레이(plug-and-play)를 지원하므로, 상기한 DAQ 장치는 특정 PC에 종속되지 않고 휴대가 가능하며, 데스크탑이나 랩탑 컴퓨터에 모두 사용할 수 있는 장점이 있다.
- [0086] 즉, 최근, 공학교육 인증 교과과정이 실험이나 설계를 중요시하고 있는 상황에서 학생들에게 집에서 실험, 실습을 가능하게 하기 위한 DAQ 장치의 통신방식으로는 USB가 최선의 선택이라고 할 수 있다.
- [0087] 여기서, 본 발명에 적용된 DAQ 장치는, 현재 시판되고 있는 여러 가지 상용 USB 장치 중에서 Cypress

Semiconductor사의 CY7C68016A를 사용하였다.

- [0088] 즉, CY7C68016A는 FX2라고 불리는데, Full speed USB 장치의 전송속도가 12Mbps인데 비하여, FX2는 High speed 를 지원하기 때문에 최대 480Mbps의 전송속도를 가진다.
- [0089] 또한, USB의 통신속도가 빠르기는 하지만, 이는 많은 큰 데이터를 하나의 프레임(frame)에 실어서 전송하는 방식이기 때문이며, 본 발명의 실시예에 따른 RCP 시스템에서는, 작은 양의 데이터를 매 샘플링 주기마다 보내고 받는 형식을 취하므로, USB 통신이 실시간 제어를 위한 데이터 전송에는 그다지 적합한 방식은 아니다.
- [0090] 하지만 Full speed USB의 프레임 레이트(frame rate)가 1ms인 것에 비해 High speed USB의 경우에는 마이크로 프레임(microframe)이라는 것을 사용하며, 그 프레임 레이트(frame rate)는 125 μ s로서 Full speed에 비해 8배 빠르고, 기존의 RS232와 같은 직렬통신에 비해서도 빠른 통신속도를 가지기 때문에, 본 발명의 실시예에 따른 RCP 시스템에서는 High speed USB 통신방식을 사용하였다.
- [0091] 또한, 도 4에 나타낸 바와 같은 다양한 입/출력 장치들 중 주요 입/출력 장치들은 본 발명자들에 의해 자체 개발된 Simulink 라이브러리 블록을 이용하여 DAQ 장치로부터 값을 받거나 보낼 수 있도록 하였다.
- [0092] 여기서, 라이브러리 블록이 지원되는 입/출력장치들의 세부사항은 다음과 같다.
- [0093] * PWM : ATMEGA128의 PWM 생성기능을 사용하며 모두 7 채널을 지원한다. 각 PWM 채널마다 ATMEGA128의 GPIO 하나씩을 할당하여 PWM의 정/역방향 정보를 나타낼 수 있다. 예를 들면, GPIO의 값이 0이고 PWM의 듀티비가 50%이면 +50%의 제어량에 해당하고, GPIO의 값이 1이고 PWM의 듀티비가 50%이면 -50%의 제어량을 나타낸다.
- [0094] * DAC : 16bit 해상도의 DAC8554를 이용하여 모두 4채널을 지원하며, 출력전압 범위는 0 ~ 5V이다.
- [0095] * Digital output : 2개의 74LS574를 이용하여 16bit의 디지털 출력(digital output)을 지원한다.
- [0096] * LED : 8bit의 정보를 On, Off 방식으로 출력할 수 있다.
- [0097] * ADC : 16bit 해상도의 ADS8344를 이용하여 모두 8채널을 지원하며, 입력전압 범위는 0 ~ 5V이다.
- [0098] * Digital input : 2개의 74LS541를 이용하여 16bit의 디지털 입력(digital input)을 지원한다.
- [0099] * Switch input : 1개의 74LS574를 이용하여 8bit의 스위치(switch) 정보를 받을 수 있다.
- [0100] * Encoder : 24bit 인코더 카운터(Encoder counter) IC인 LS7566를 2개 이용하여 8개의 incremental encoder를 처리할 수 있다.
- [0101] 계속해서, 도 5 및 도 6을 참조하여, 본 발명의 실시예에 따른 RCP 시스템의 통신 블록 라이브러리의 구체적인

내용에 대하여 설명한다.

- [0102] 즉, 본 발명의 실시예에 따른 RCP 시스템은, Simulink를 통해 구성한 제어기 모델이 제어 연산을 수행하는 구조이며, 제어 연산에 필요한 센서 데이터는 DAQ 장치를 통해 측정된 후 USB 통신을 통하여 PC로 전송된다.
- [0103] 또한, 제어 연산의 결과로 얻어진 제어 데이터는 USB 통신을 통하여 DAQ 장치로 전송되고 DAQ 장치에 의해 해당하는 출력 장치를 통하여 출력된다.
- [0104] 아울러, 본 발명의 실시예에 따른 RCP 시스템은, USB 통신을 통하여 DAQ 장치로부터 센서 데이터를 전송받거나, 제어 데이터를 DAQ 장치로 전송하기 위해 Simulink 환경 내에서 사용할 수 있는 10개의 통신 라이브러리 블록을 지원한다.
- [0105] 즉, 도 5를 참조하면, 도 5는 본 발명의 실시예에 따른 RCP 시스템의 Simulink용 통신 라이브러리 블록을 나타내는 도면이다.
- [0106] 더 상세하게는, 도 5에 나타난 바와 같이, 2열로 구성된 라이브러리 블록 중에서 첫 번째 열에 위치한 블록들은 DAQ 장치로부터 데이터를 전송받기 위한 수신(Receive) 블록이고, 두 번째 열에 위치한 블록들은 DAQ 장치 쪽으로 데이터를 전송하기 위한 송신(Send) 블록들이다.
- [0107] 또한, 도 5에 나타난 각각의 블록 중에서 숫자가 병기된 블록들은 여러 개의 채널을 지원하는 블록을 나타내며, 도 5에 나타난 실시예에 있어서는, 모두 1번 채널이 선정된 경우의 블록들의 모양을 나타내고 있다.
- [0108] 아울러, 이와 같이 여러 개의 채널을 동시에 사용하고자 할 경우에는, 도 6에 나타난 바와 같이, 블록을 더블클릭하여 나타나는 대화상자를 통해 사용하고자 하는 채널을 지정할 수 있도록 구성할 수 있다.
- [0109] 즉, 도 6을 참조하면, 도 6은 대화상자를 통하여 채널의 지정 및 블록의 형태를 변경하는 화면을 나타내는 도면으로, ADC 블록의 1번과 3번 채널의 정보를 USB 통신을 이용하여 수신하도록 대화상자를 설정하는 것과, 대화상자 설정 후에 ADC 블록의 출력포트가 2개로 변한 것을 보여주고 있다.
- [0110] 계속해서, 도 7 내지 도 9를 참조하여, 상기한 바와 같은 본 발명의 실시예에 따른 RCP 시스템을 이용하여 제어기를 구성하는 내용에 대하여 설명한다.
- [0111] 본 발명자들은, 상기한 바와 같은 본 발명의 실시예에 따른 RCP 시스템을 적용하여 도 7에 나타난 바와 같은 독립진자 시스템에 대한 스윙업 제어실험을 수행하였다.
- [0112] 즉, 도 7을 참조하면, 도 7은 본 발명의 실시예에 따른 RCP 시스템을 이용하여 독립진자의 실시간 제어 실험을 수행하기 위해 자체 개발한 독립진자의 실물 사진을 나타내는 도면이다.
- [0113] 여기서, 실험에 사용된 독립진자 시스템은, 봉이 직립상태를 기준으로 이루는 각과, 카트(cart)의 수평방향 변위를 측정하기 위해 2개의 인크리멘탈 인코더(Incremental encoder)를 센서로 사용하며, 카트를 좌우로 이동시

키기 위한 구동기(actuator)로서 DC 모터(DC motor)를 사용하고 있다.

- [0114] 또한, DC 모터는 H-브리지(H-bridge) 형태의 모터 드라이버(motor driver)에 의해 구동되며, 따라서 독립진자 시스템을 제어하기 위해서는 도 5에 나타난 라이브러리 블록 중에서 인코더 블록(ENC)과 PWM 블록을 사용한다.
- [0115] 여기서, 인코더 블록의 경우, 2개의 채널을 처리해야 하므로 대화 상자를 이용하여 해당되는 채널을 사용하도록 설정한다.
- [0116] 즉, 도 8을 참조하면, 도 8은 본 발명의 실시예에 따른 RCP 시스템에서 제공되는 통신 라이브러리 블록(인코더 블록과 PWM 블록)과, Simulink에서 제공하는 기능 블록들을 조합하여 구성한 독립진자 시스템의 스윙업 제어기 모델의 전체적인 구성을 개략적으로 나타내는 도면이다.
- [0117] 더 상세하게는, 도 8에 나타난 바와 같이, 인코더 블록의 1번 채널과 2번 채널이 활성화 되어 있는 것을 볼 수 있으며, 제어 연산 자체가 Simulink 내에서 이루어지고 센서 데이터도 USB 통신을 통해서 Simulink 쪽으로 전송되므로, 모든 신호를 Simulink의 Scope 블록을 이용하여 실시간으로 모니터(monitor) 할 수 있으며, 또한, 저장하고 싶은 데이터를 Matlab의 Workspace로 저장할 수도 있다.
- [0118] 아울러, 도 8에 나타난 제어모델의 샘플링 주기는 8ms로, 즉, 1초당 125회의 제어연산을 수행하도록 설정된다.
- [0119] 이때, 샘플링 주기는 해당 Simulink 모델의 샘플링 주기와 동일하므로, Simulink 모델의 파라미터 설정을 통해 샘플링 주기를 설정할 수 있다.
- [0120] 도 9를 참조하면, 도 9는 본 발명의 실시예에 따른 RCP 시스템을 이용하여 도 7에 나타난 독립진자의 스윙업 제어실험을 수행한 실험결과를 나타내는 도면이다.
- [0121] 즉, 도 9에 나타난 바와 같이, 처음에 지면을 향하도록 위치해 있던 봉이 점점 흔들리면서 7초쯤 독립 상태로 안정화되는 것을 볼 수 있다.
- [0122] 여기서, 도 9는, 도 7에 나타난 모델에서, theta라고 표시된 신호에 Simulink의 To Workspace 블록을 활용하여 데이터를 저장한 후, Matlab의 stairs 명령을 이용하여 도시한 것이다.
- [0123] 즉, 상기한 바와 같이, 본 발명의 실시예에 따른 RCP 시스템을 이용하면, 실시간 제어 실험과 더불어 실시간 모니터링이 가능하고, 데이터를 저장하여 오프라인으로 처리할 수도 있다.
- [0124] 계속해서, 도 10 내지 도 13을 참조하여, 본 발명의 실시예에 따른 RCP 시스템의 소프트웨어 구현에 대하여 설명한다.
- [0125] 먼저, 본 발명의 실시예에 따른 RCP 시스템에서 지원되는 10개의 통신 라이브러리 블록은 S-Function을 이용하여 구현하였다.

- [0126] 더 상세하게는, S-Function은 Simulink에서 기본적으로 지원하는 라이브러리 블록을 조합해서는 구현할 수 없는 블록을 사용자가 정의하여 사용할 수 있도록 하는 기능으로써, 블록의 동작을 Matlab 파일(file) 또는 C-code로 기술하도록 구성되어 있다.
- [0127] 또한, S-Function을 C-code를 이용하여 정의하면 실행속도를 크게 높일 수 있고, 하드웨어 장치를 제어할 수 있는 장점이 있으므로, 본 발명의 실시예에 따른 RCP 시스템에서는 C-code 기반의 S-Function을 사용하였다.
- [0128] 아울러, 이러한 C-code 기반의 S-Function 작성은, 미리 지정된 이름의 callback function들을 정의하는 것으로 구성된다.
- [0129] 즉, 통신 라이브러리 블록을 구현하기 위해, mdlInitializeSizes, mdlInitializeSampleTimes, mdlStart, mdlOutputs, mdlTerminate와 같은 이름의 callback function을 각각 S-Function에 정의하였다.
- [0130] 여기서, S-Function에 기술된 callback function들은 사용자가 호출하는 것이 아니라 Simulink가 적절한 순간에 호출하게 되며, 적절한 순간에 호출된다는 의미를 이해하기 위해서는 Simulink의 동작방식에 대한 이해를 필요로 한다.
- [0131] 구체적으로는, 도 10을 참조하면, 도 10은 Simulink가 simulation을 수행하는 방식을 나타내는 흐름도이며, 또한, 도 11을 참조하면, 도 11은 Simulink의 동작방식을 설명하기 위한 Simulink 모델의 구성예를 나타내는 도면이다.
- [0132] 즉, 도 11과 같은 간단한 Simulink 모델을 상정하면, 도 11의 모델을 수행하면, Simulink에서는 도 10에 나타난 흐름도에 따라 단계별로 시뮬레이션을 수행한다.
- [0133] 여기서, 처음 진행되는 단계는 모델 초기화 단계로, 모델을 실행하기 위해 필요한 여러 가지 초기화 작업을 수행한다.
- [0134] 이러한 초기화 작업 중 하나는, 모델에 포함된 블록들의 계산 실행순서(execution order)를 결정하는 과정이다.
- [0135] 도 10에 주어진 모델의 실행순서를 Simulink를 이용하여 확인해 보면, Integrator -> SineWave -> Gain -> Scope의 순서를 가지는 것을 알 수 있다.
- [0136] 또한, 초기화 단계가 종료되면, 시뮬레이션 루프(simulation loop) 내로 진입하여 도 10에 주어진 순서대로 단계별 계산을 수행한다.
- [0137] 먼저, 블록의 출력을 계산하는 단계를 수행하며, 이때, Simulink는 초기화 단계에서 구한 실행순서에 따라 각 블록의 출력 계산 함수를 호출하게 된다.
- [0138] 따라서 Integrator의 출력 계산 함수가 제일 먼저 호출되고, SineWave, Gain, Scope 블록의 순서로 각 블록의 출력 계산 함수가 호출된다.

- [0139] 출력계산 단계가 종료되면, Simulink는 이산형 상태의 갱신 단계와 연속형 상태의 갱신 단계를 차례로 수행하며, 이러한 단계들 역시 초기화 단계에서 구한 실행순서에 따라 수행되나, 블록이 이산형 상태나 연속형 상태를 가졌을 때만 수행된다.
- [0140] 만약, S-Function을 이용하여 구성된 블록이 Simulink 모델에 사용되었다면, 해당 S-Function 블록도 모델 전체 내에서 특정 실행순서를 가지게 된다.
- [0141] 즉, Simulink는, 출력계산 단계에서 S-Function 블록의 순서가 되었을 때 S-Function 블록의 출력을 계산하는 callback function인 mdlOutputs를 호출하게 된다.
- [0142] 물론, S-Function 블록이 이산형 상태나 연속형 상태를 가질 경우, 그 블록의 실행순서가 되었을 때 상태를 갱신하기 위한 callback function이 호출되게 된다.
- [0143] 또한, S-Function 블록은 모델 내에서 여러 개의 인스턴스(instance)를 가질 수 있고, 즉, 같은 S-Function을 사용하는 블록을 여러 개 사용할 수 있으며, 각각의 블록들은 대화상자를 통해 다른 파라미터를 전달받을 수 있다.
- [0144] 예를 들면, S-Function을 이용하여 PID 제어 블록을 구성하였다면, 모델 내에 여러 개의 PID 블록을 사용할 수 있고, 각각의 PID 블록들은 각기 고유의 파라미터(예를 들면, P 계수, I 계수, D 계수)를 가질 수 있다.
- [0145] 본 발명의 실시예에 따른 RCP 시스템에서 제공하는 10개의 통신 라이브러리 블록들은 하나의 S-Function을 이용하여 정의되며, 각 통신 블록의 기능 구별은 내부적으로 전달받는 파라미터를 통해 결정된다.
- [0146] 예를 들면, PWM 블록은 '0x80'을, ADC 블록은 '0x20'을 파라미터로서 전달받는다.
- [0147] 물론, 이러한 파라미터를 전달하는 과정에 일반 사용자들은 개입할 필요가 없고, 단지 해당 라이브러리 블록을 모델 내에 포함시키기만 하면 된다.
- [0148] 통신 라이브러리 블록을 구현하는데 있어서 중점을 되는 점은 데이터를 일괄적으로 전송하고 일괄적으로 수신하도록 처리하는 것이다.
- [0149] 만약, 하나의 Simulink 모델에 여러 개의 Send 블록과 Receive 블록이 사용될 경우, USB를 통한 데이터의 전송이 해당 통신 블록의 실행 순서가 되었을 때 이루어지는 방식으로 처리한다면, 하나의 통신 블록의 데이터를 처리하기 위해 USB에서는 하나의 마이크로프레임이 소요된다.
- [0150] 따라서 통신 블록이 여러 개일 경우, 통신에 소요되는 시간이 늘어나게 되고, 결국, 제어시스템의 샘플링 주기를 짧게 하는데 저해 요소가 된다.

- [0151] 계속해서, 도 12를 참조하여, 이러한 문제를 더욱 상세하게 설명한다.
- [0152] 즉, 도 12를 참조하면, 도 12는 본 발명의 실시예에 따른 RCP 시스템에서 제공하는 10개의 통신 라이브러리 블록을 모두 사용하는 경우의 동작을 테스트하기 위해 구성한 Simulink 모델이다.
- [0153] 도 12에 있어서, 각 블록의 우측 상단에 보이는 숫자는 블록의 실행순서이며, 예를 들면, ADC 블록은 실행순서 0, 인코더 블록은 실행순서 19번을 가진다.
- [0154] 또한, 실행순서가 보이지 않는 블록은 여러 개의 블록으로 구성된 서브시스템이다.
- [0155] 만약, 데이터 통신을 일괄 전송방식이 아닌 블록별 전송방식을 택한다면, ADC 블록은 0번 실행순서 때에 데이터를 수신받고 Buzzer 블록은 4번 실행순서에 DAQ 장치 쪽으로 데이터를 전송하게 된다.
- [0156] 따라서 이러한 방식은, 모두 10개의 통신블록이 있으므로 10개의 마이크로프레임이 소요되며, 이는 1.25ms에 해당한다.
- [0157] 그러나 이러한 시간은 가장 이상적인 상황을 가정했을 경우에 USB 통신에만 소요되는 시간이며, 실제로는, 모델 내의 다른 블록들을 계산하는데 소요된 시간을 포함하고 윈도우(Windows) 운영체제가 멀티태스킹(multitasking)인 점을 고려하면, 모델 내의 모든 블록을 한번 계산하는데 소요되는 시간은 이것보다 훨씬 더 길어져 샘플링 주기가 5ms 또는 10ms가 넘는 경우에만 적용할 수 있게 될 것이다.
- [0158] 따라서 본 발명자들은, 이러한 문제점을 해결하고 빠른 샘플링 주기를 가지는 제어 시스템에도 사용할 수 있도록, 일괄 전송방식을 지원하도록 S-Function의 callback function을 구현하였다.
- [0159] 계속해서, S-Function에 구현한 5개의 callback function 중에서 가장 중요한 mdlStart와 mdlOutput을 중심으로 S-Function에 기술된 callback function의 동작을 설명한다.
- [0160] 통신 블록을 사용했을 경우, 일괄적으로 센서 데이터를 DAQ 장치로부터 수신 받고 일괄적으로 제어 데이터를 DAQ 장치로 전송하기 위해서는, 먼저 어떠한 데이터가 통신의 대상이 되는지에 대한 정보를 DAQ 장치 측으로 넘겨주어야 한다.
- [0161] 이러한 정보는 2개의 테이블(table)(code 상에서는 array로 구현된다)을 통해서 전달되는데, 하나는 RxDataInfo table이고, 또 하나는 TxDataInfo table이다.
- [0162] 여기서, RxDataInfo table은 DAQ 장치로부터 PC 측으로 전송해야할 데이터의 종류를 순서대로 가지고 있으며, TxDataInfo table은 PC 측에서 DAQ 장치로 전송해야할 데이터의 종류를 순서대로 가지고 있다.
- [0163] 예를 들면, 도 12에 나타난 모델에 대한 테이블은 도 13과 같이 나타내진다.
- [0164] 즉, 도 13을 참조하면, 도 13은 도 12에 나타난 Simulink 모델에 대한 RxDataInfo 및 TxDataInfo 테이블의 구

성을 나타내는 도면이다.

- [0165] 도 13에 있어서, 도 13에 기재된 데이터의 순서는 Simulink 모델에 포함된 통신 블록들의 실행순서(execution order)를 따르는 것을 알 수 있다.
- [0166] 구체적으로는, Simulation loop가 실행되기 전에 DAQ 장치는 이러한 정보를 PC(Simulink)로부터 전달받게 되고, Simulation loop가 시작되면 매번 timer interrupt service가 시행될 때마다 RxDataInfo table에 있는 정보를 이용하여 해당 데이터를 모두 측정한 후, 일괄적으로 USB 통신을 이용하여 PC 측으로 전송하고, PC측의 Simulink가 제어 연산을 수행하고난 결과를 일괄적으로 DAQ 장치에 전송하면, TxDataInfo table에 있는 정보를 이용하여 수신된 데이터를 해당하는 출력장치에 출력하게 된다.
- [0167] 즉, 통신 블록을 정의하는 S-Function에 정의되어 있는 mdlStart라는 callback function에서 수행하는 여러 작업중 가장 중요한 2가지 작업은, 바로 이 2개의 table을 생성하는 작업과, 모델 내에 사용된 통신 블록 중에서 실행순서가 가장 빠른 블록과 가장 늦은 블록의 종류를 알아내는 작업이다.
- [0168] 여기서, 도 12에 나타낸 모델의 경우, ADC 블록이 통신 블록 중에서 실행순서가 가장 빠르고, Switch 블록이 실행순서가 가장 늦은 블록이다.
- [0169] 또한, Simulink가 모델 초기화 작업 단계를 수행할 때 모델 내 블록들의 실행순서에 따라 각 블록의 초기화 함수를 호출하게 되는데, RCP용 통신블록의 경우에는 mdlStart가 호출된다.
- [0170] 따라서 각 통신블록의 mdlStart가 호출되면, 각 통신블록이 다루는 데이터 정보를 전역변수로 잡혀있는 RxDataInfo 또는 TxDataInfo table에 순서대로 채워 넣게 된다.
- [0171] 아울러, 통신블록의 mdlOutput이라는 callback function에서는 다음과 같이 5가지의 조건을 순서대로 검사하고, 조건을 만족할 경우 할당된 작업을 수행한다.
- [0172] 여기서, 이하의 5가지 조건은 상호 배타적인 조건이 아니므로, 2개 이상의 조건이 만족 될 수 있다.
- [0173] (1) $t = 0$ 이고 통신블록 중 실행순서가 가장 앞서는 블록일 경우
- [0174] mdlStart를 통해 구한 table 정보를 DAQ 장치 측으로 넘겨주고, DAQ 장치의 timer interrupt 기능을 활성화시킨다.
- [0175] 이 단계에서, DAQ 장치는 PC로부터 넘어온 정보를 이용하여 PC로 전송해야할 데이터의 종류와 PC로부터 전송받아야할 데이터의 종류를 알 수 있게 된다.
- [0176] 또한, DAQ 장치의 timer interrupt가 활성화되므로, PC로 전송해야 할 센서 데이터를 모두 측정한 후에 PC측으로 일괄 전송하게 된다.
- [0177] (2) 통신블록 중 실행순서가 가장 앞서는 블록일 경우

- [0178] 해당 블록이 Send 블록이든 Receive 블록이든 상관없이, 무조건 DAQ 장치로부터 USB 통신을 통해 일괄 전송된 센서 데이터를 하나의 배열 RxBuffer에 저장한다.
- [0179] 도 12의 모델의 경우는, ADC 블록이 이러한 경우에 해당한다.
- [0180] (3) Receive 블록일 경우
- [0181] RxBuffer에 저장된 일괄 센서 데이터 중에서 해당 Receive 블록에 해당되는 데이터를 찾아 Receive 블록의 출력을 갱신한다.
- [0182] (4) Send 블록일 경우
- [0183] 블록의 입력 데이터를 TxBuffer라는 배열에 채워 넣는다.
- [0184] 여기서, Send 블록이 여러 개 있을 경우, 배열에는 여러 개의 데이터가 채워지게 된다.
- [0185] (5) 통신블록 중 실행순서가 가장 낮은 블록일 경우
- [0186] 해당 블록이 Send 블록이든 Receive 블록이든 상관없이 무조건 TxBuffer에 저장된 데이터를 USB 통신을 이용하여 DAQ 장치 측으로 전송한다.
- [0187] 도 12의 모델의 경우, Switch 블록이 이러한 경우에 해당한다.
- [0188] 또한, DAQ 장치에서는, 이 단계에서 넘어온 일괄 데이터를 TxDataInfo table에 있는 정보에 맞추어 해당 출력장치로 데이터를 출력한다.
- [0189] 따라서 본 발명의 실시예에 따른 RCP 시스템은, 상기한 방법을 통하여 하나의 Simulink 모델에 여러 개의 통신블록이 있어도 일괄전송과 일괄수신이 가능하도록 구성된다.
- [0190] 아울러, 상기한 바와 같은 방법을 통하여 구성되는 RCP 시스템은, 샘플링 주기가 1ms 정도인 제어 시스템에도 안정적으로 적용할 수 있다.
- [0191] 다음으로, 상기한 바와 같이 하여 구현된 본 발명의 실시예에 따른 RCP 시스템에 대하여 실시간 성능시험을 수행한 결과에 대하여 설명한다.
- [0192] 즉, 일체의 실시간 연산을 내장형 연산 장치에서 수행하는 기존의 RCP 시스템과는 달리, 본 발명의 실시예에 따른 RCP 시스템의 경우는, 제어연산 자체는 고속 계산이 가능한 PC에서 이루어지고 있기 때문에 제어연산에 소요되는 계산량 자체는 제어시스템의 샘플링주기 선정에 그다지 영향을 미치지 않는다.
- [0193] 따라서 본 발명의 실시예에 따른 RCP 시스템에서, 샘플링 주기 선정에 영향을 주는 요소는 USB 통신에 의해 소요되는 시간이다.

- [0194] 이 때문에, 본 발명의 실시예에 따른 RCP 시스템에서는, USB 통신에 소요되는 시간을 줄이기 위해 일괄 전송방식을 제안하였으나, 그렇다고 하더라도, 기존 방식의 RCP 시스템에서 가능한 것처럼 수십 KHz에서 수백 KHz에 이르는 고속 샘플링은 불가능하며, 물론, 기존 RCP 시스템에서도 그 정도의 고속 샘플링은 제어 연산량이 비교적 작은 경우거나 정수형 연산만을 사용한 경우에만 가능하다.
- [0195] 따라서 이하에서는, 본 발명의 실시예에 따른 RCP 시스템을 이용할 경우 제어 시스템의 샘플링 주기를 어느 정도까지 낮출 수 있는지에 대하여 고찰한 결과에 대하여 설명한다.
- [0196] 일반적으로, 고속(High speed) USB를 사용할 경우, 풀스피드(Full speed) USB에 비해 프레임(frame) 속도가 8배 빠르고, 또한, High speed USB에서는 마이크로프레임(microframe)을 사용하며, 하나의 마이크로프레임은 125 μ s 동안 지속된다.
- [0197] 따라서 DAQ 장치로부터 PC로의 데이터 전송이 하나의 마이크로프레임 내에 이루어지고 PC로부터 DAQ로의 데이터 전송이 또 하나의 마이크로프레임에 이루어진다면, USB 통신에 소요되는 시간은 가장 이상적인 경우 250 μ s 이내가 될 수 있다.
- [0198] 즉, PC에서 이루어지는 제어 연산이 하나의 마이크로프레임 내에 이루어진다면 통신과 제어연산을 포함한 모든 시간이 375 μ s 보다 작을 수 있다.
- [0199] 만약, PC에서 이루어지는 제어 연산의 연산량이 2개의 마이크로프레임 정도 이내에서 이루어진다면, 전체 소요 시간은 500 μ s 보다 작을 수 있다.
- [0200] 따라서 이러한 분석에 근거하면 2KHz 정도의 샘플링 주파수도 가능할 수 있으며, 아울러, 이러한 분석이 실제 시스템의 제어 실험에서도 적용할 수 있는 합리적인 분석인지를 살펴보기 위해 상기한 도립진자의 제어 실험을 예로 들어 설명한다.
- [0201] 먼저, 도립진자 제어 시스템의 샘플링 주기는 8ms이고, 따라서 샘플링 주파수는 125Hz가 되며, 한 번의 루프 계산이 완전히 종료되는데 소요되는 시간을 측정하기 위하여 ATMEGA128의 timer interrupt가 시작되는 시점에 포트(Port) D의 2번 포트에 logic 1을 출력하고 timer interrupt 안에서 지정한 작업이 모두 종료되는 순간에 역시 Port D의 2번 포트에 logic 0을 출력하였다.
- [0202] 그리고 실험이 진행되는 동안 해당 포트의 신호값을 디지털 오실로스코프를 이용하여 측정한 후 저장한 데이터를 Matlab을 이용하여 도시하였다.
- [0203] 즉, 도 14를 참조하면, 도 14는 상기한 바와 같이 샘플링 주기를 8ms로 하여 도 12에 나타낸 RCP 시스템의 실시간 수행 성능을 시험한 결과를 나타내는 도면이다.
- [0204] 도 13에 있어서, T는 샘플링 주기(timer interrupt의 주기와 같음)이고, T1은 timer interrupt에 진입했다가 빠져나오는데 걸리는 시간이며, T2는 아무것도 하지 않고 다음 timer interrupt에 진입할 때까지 기다리는 시간이고, 따라서 당연히 $T = T1 + T2$ 가 성립한다.

- [0205] 도 14를 통해 알 수 있는 바와 같이, 샘플링 주기를 8ms로 하여 실시간 제어 실험을 수행하였을 때에는, T1이 T에 비해 현저히 작으므로, 여전히 시간적 여유가 많이 있는 것을 볼 수 있다.
- [0206] 따라서 도립진자 시스템의 경우 샘플링 주기를 8ms보다 더 작게 선정해도 실시간 성능에 문제가 없을 것이라는 것을 예상할 수 있다.
- [0207] 이에 대하여 본 발명자들은, 샘플링 주기를 1ms로 낮게 조정한 후 제어실험을 수행하고 실시간 성능을 오실로스코프로 확인하여 그 결과를 도 15에 나타내었다.
- [0208] 즉, 도 15를 참조하면, 도 15는 상기한 바와 같이 샘플링 주기를 1ms로 하여 도 12에 나타낸 RCP 시스템의 실시간 수행 성능을 시험한 결과를 나타내는 도면이다.
- [0209] 도 15에 있어서, 각각의 숫자는 timer interrupt에 진입했다가 빠져나오는 데까지 소요된 시간을 μs 단위를 이용하여 나타낸 것이다.
- [0210] 도 15에 나타낸 바와 같이, 소요된 시간이 모두 동일하지는 않지만 비교적 균일하게 나타나는 것을 볼 수 있으며, 가장 길게 소요된 시간도 $500\mu s$ 를 넘지 않는 것을 볼 수 있다.
- [0211] 따라서 상기한 바와 같은 결과에 근거하면, 2KHz까지 샘플링 주기를 높이는 것도 가능하다.
- [0212] 하지만 윈도우(Windows) 환경이 멀티태스킹(multitasking)을 지원하고 실시간 운영체제가 아닌점을 고려한다면, 본 발명의 실시예에 따른 RCP 시스템을 이용하여 실시간 제어실험을 수행할 경우, 바람직하게는, 1KHz 정도의 샘플링 주기가 적합하다.
- [0213] 또한, 실제로 이러한 1KHz 정도의 샘플링 속도로 제어실험을 해볼 수 있는 시스템은 상당히 많기 때문에, 본 발명의 실시예에 따른 RCP 시스템은 학부나 대학원 과정의 실험을 수행하는 데 무리 없이 사용될 수 있다.
- [0214] 계속해서, 상기한 바와 같은 본 발명의 실시예에 따른 RCP 시스템의 알고리즘 및 소프트웨어 구현에 대하여 보다 상세하게 설명한다.
- [0215] 즉, 본 발명의 실시예에 따른 RCP 시스템에서 사용되는 소프트웨어는 PC 측의 Simulink에서 수행되는 S-Function(AvrDaqS-fun.cpp)과 DAQ 장치 측에서 수행되는 펌웨어(firmware)(AvrDaqTarget.c)로 구성된다.
- [0216] 여기서, AvrDaqSfun.cpp는 Matlab의 mex command를 이용하여 컴파일하며, 결과물로 얻게 되는 MEX(Matlab Executable) 파일의 이름은 AvrDaqSfun.dll이나, Matlab 7.1 이후에는 MEX file의 확장자가 mexw32로 변경되었으므로(32-bit version Windows의 경우) Matlab 7.1 이후의 버전을 이용할 경우 MEX file의 이름은 AvrDaqSfun.mexw32가 된다.
- [0217] 확장자에서 알 수 있듯이, 생성되는 파일은 dll(dynamic link library)이며, AvrDaqSfun.dll은 Simulink를 실행시킬 경우 Simulink에 의해 호출되어 사용된다.

- [0218] 본 발명의 실시예에 따른 RCP 시스템에서 제공하는 10개의 통신블록(이하, 설명을 간결하게 하기 위해 'RCP 블록'이라 한다)은, Time, ADC, ENC, DI, Switch의 5개의 블록을 포함하는 수신블록(Receive block)과, PWM, DAC, DO, Buzzer, LED의 5개의 블록을 포함하는 송신블록(Send block)으로 구성되어 있으며, 이들 블록은 모두 S-Function 블록으로 구현되어 있다.
- [0219] 여기서, Receive는 DAQ장치로부터 PC 측으로 데이터를 전송받는 것을, Send는 PC 측에서 DAQ 장치로 데이터를 전송하는 것을 나타낸다.
- [0220] 10개의 RCP 블록이 각각 다른 기능을 수행하지만, 즉, 다른 종류의 데이터를 송/수신하나, 이들 RCP 블록이 사용하는 dll 파일은 AvrDaqSfun.dll로 모두 동일하며, 단지 넘겨받는 파라미터에 따라 다른 기능을 수행하는 블록으로 동작하도록 설계된다.
- [0221] 즉, 도 16을 참조하면, 도 16은 RCP 블록에 사용된 S-Function 블록의 대화상자를 나타내는 도면이다.
- [0222] 도 16에 있어서, 1로 표시된 부분에 S-Function의 이름인 AvrDaqSfun이, 그리고 2로 표시된 부분에는 S-Function에서 사용하는 파라미터인 BlockType, SampleTimeInfo, ReceiveDataInfo, SendDataInfo가 사용되었다.
- [0223] 이들 4개의 파라미터 중에서 첫 번째 파라미터인 BlockType의 값에 의해 RCP 블록의 종류가 결정된다.
- [0224] 모든 RCP 블록은 마스킹(Masking) 되어 있어 종류에 따라 블록의 외형이 다르게 되어 있으며, 예를 들면, ENC, PWM, ADC, DAC 등과 같이, 여러 개의 채널을 지원하는 블록의 경우는, 대화상자를 통해 사용하고자 하는 채널을 지정할 수 있도록 구성되고, 이를 다른 Simulink 블록과 적절히 연결하여 원하는 기능을 얻도록 모델을 구성한다.
- [0225] 또한, 각각의 RCP 블록들이 Simulink로부터 넘겨받는 4개의 파라미터는 다음과 같다.
- [0226] BlockType : 블록의 종류를 나타내는 넘버(number)
- [0227] SampleTimeInfo : Sampling time에 대한 정보
- [0228] ReceiveDataInfo : Receive block일 경우의 채널 정보
- [0229] SendDataInfo : Send block일 경우의 채널 정보
- [0230] 각각의 RCP 블록은 고유의 블록 타입 넘버(block type number)를 가지며 S-Function은 이 block type number를 스위치 스테이트먼트(switch statement)를 이용하여 구분함으로써 각 블록이 서로 다른 기능을 수행하도록 한다.
- [0231] 즉, 도 17을 참조하면, 도 17은 RCP 블록의 종류와 Block Type Number를 정리하여 표로 나타낸 도면이다.

- [0232] 또한, 도 18을 참조하면, 도 18은 ReceiveDataInfo와 SendDataInfo의 내용을 표로 정리하여 나타낸 도면이다.
- [0233] 도 18에 나타낸 바와 같이, ReceiveDataInfo와 SendDataInfo는 멀티 채널을 지원하는 RCP 블록의 경우 사용되는 채널정보를 담고 있고, 멀티 채널이 아닌 경우에는 block type number의 값을 가진다.
- [0234] 예를 들면, 1번과 3번 채널을 사용하도록 설정한 Encoder block의 경우, 파라미터 값은 다음과 같게 된다.
- [0235] BlockType = 0x10, ReceiveDataInfo=[0x10;0x12], SendDataInfo=[]
- [0236] 여기서, Encoder block은 2개의 채널을 사용하므로, ReceiveDataInfo는 벡터가 되며, 또한, Encoder block은 Receive block에 속하므로, SendDataInfo는 빈 행렬(blank matrix)이 된다.
- [0237] 아울러, 하나의 Simulink model에는 같은 종류의 RCP 블록이 2개 이상 사용될 수 없으며, 만약, 이 규칙을 지키지 않고 Simulink model을 구성하면 실행시 오류 메시지가 발생한다.
- [0238] 또한, 4개의 파라미터 중에서 SampleTimeInfo는 Simulink model에 설정된 sample time을 이용하여 ATMEGA128의 timer interrupt의 주기를 설정하기 위한 설정 정보를 담고 있다.
- [0239] 즉, 본 발명의 실시예에 따른 RCP 시스템은 ATMEGA128이 가지고 있는 4개의 Timer 중에서 8bit timer인 Timer 2를 사용하며, 참고로, Timer 1(16bit), Timer 3(16bit), Timer 4(8bit)는 PWM의 생성에 사용된다.
- [0240] Timer 2를 이용할 경우, timer interrupt의 주기 ts는 다음과 같이 결정된다.
- [0241] [수학식 1]
- $$ts = \frac{N(1+OCR2)}{16 \times 10^6}$$
- [0242]
- [0243] 여기서, N은 분주비이고, OCR2는 timer interrupt 설정과 관련된 레지스터(register)(OCR2)에 설정된 값이다.
- [0244] SampleTimeInfo를 이용하여 N과 OCR2의 값을 넘겨주면, Simulink와 DAQ 장치가 통신을 초기화하는 과정에서 이 정보가 DAQ 장치로 전송되어 ATMEGA128의 timer interrupt 주기를 설정하게 된다.
- [0245] 따라서 SampleTimeInfo는 개념상으로 다음과 같은 구조를 가진다.
- [0246] SampleTimeInfo = [N;OCR2]

- [0247] ATMEGA128의 timer interrupt의 주기를 t_s 로 설정하기 위해서는, 분주비 N과 OCR2를 적당하게 지정해 주어야 하며, N과 OCR2를 설정하는 방법은 다음과 같다.
- [0248] 즉, ATMEGA128은 16MHz 클럭을 사용하므로, 1초에 16×10^6 개의 클럭 펄스(clock pulse)가 발생하며, 따라서 t_s 의 시간 동안에 x만큼의 클럭 펄스가 발생한다고 하면,
- [0249]
$$X = t_s \times 16 \times 10^6$$
- [0250] 예를 들면, $t_s = 1\text{ms}$ 라고 하면, 1ms 동안에는 $x = 16,000$ 개의 클럭 펄스가 발생하며, Timer/Counter의 분주비를 어떻게 설정하면 16,000개의 클럭 펄스가 발생하는 동안 8bit 타이머의 카운트 레지스터(count register)가 꽉 차게(full) 되는가에 대한 답은 다음과 같다.
- [0251]
$$x/2^8 = 16000/2^8 = 62.5$$
- [0252] 즉, 분주비를 $N = 62.5$ 로 설정하면 클럭 펄스가 62.5개 발생할 때 분주회로가 하나의 펄스를 발생하고, 이 펄스를 받아 동작하는 8bit 카운터가 오리지널 클럭(original clock)이 16,000개의 펄스를 발생할 동안 꽉 차게 된다.
- [0253] 여기서, 분주비는 정수로 설정해야 하며, 또한, ATMEGA128의 경우 Timer 2에 사용할 수 있는 분주비는 1, 8, 64, 256, 1024의 값 중 하나만 사용할 수 있다.
- [0254] 즉, 상기한 62.5는 최소 분주비이므로, 이보다 큰 분주비를 사용해야만 8bit 타이머의 카운트 레지스터가 꽉 차게 되지 않는다.
- [0255] 따라서 선택할 수 있는 분주비는 $N = 64, 256, 1024$ 가 되며, x와 N이 주어지면 OCR2는 다음과 같이 정해야 한다.
- [0256]
$$\text{OCR2} = \text{round}(x/N) - 1$$
- [0257] 여기서, 이와 같이 OCR2를 정했을 때 지정하고자 하는 timer interrupt의 주기 t_s 가 정확히 나오지 않을 수 있다.
- [0258] 즉, 분주비는 원하는 값을 임의로 사용하는 것이 아니라 지정된 정수값 중의 하나를 사용해야 하기 때문이다.
- [0259] 따라서 [수학식 1]을 이용하여 계산했을 때 지정하고자 하는 timer interrupt의 주기 t_s 와 가장 가까운 값을 리턴(return) 해주는 N과 OCR2를 사용해야 한다.
- [0260] 즉, $N = 64, 256, 1024$ 일 경우, $\text{OCR2} = 249, 62, 15$ 가 되며, 따라서 $(N, \text{OCR2}) = (64, 249), (256, 62),$

(1024, 15)의 3개의 쌍이 후보로 상정될 수 있다.

[0261] N과 OCR2를 이용하여 설정한 timer interrupt의 주기와 실제로 설정하고자 했던 주기 ts와의 차이는 다음과 같이 계산될 수 있다.

$$[0262] \quad error = \left| ts - \frac{N(1+OCR2)}{16 \times 10^6} \right|$$

[0263] 여기서, 3개의 쌍에 대해 위의 값을 계산해 보면,

[0264] error of (64, 249) = 0

[0265] error of (256, 62) = 8×10^{-6}

[0266] error of (1024, 15) = 2.4×10^{-5}

[0267] 따라서 ts = 1ms로 설정하고자 할 경우 N = 64, OCR2 = 249로 설정하면 오차가 없이 정확하게 timer interrupt의 주기를 설정할 수 있다.

[0268] 하지만 timer interrupt의 주기를 원하는 값으로 설정할 수 없을 수도 있으며, 예를 들면, ts = 9ms로 설정하고자 하는 경우, 위에서 설명한 과정을 적용하여 N과 OCR2를 설정하면, 실제 설정되는 timer interrupt의 주기는 9.024ms로써 약 0.27% 정도의 차이를 보인다.

[0269] 또한, 설정할 수 있는 timer interrupt의 주기의 최대값 또한 제약이 따르며, 이 값은 N = 1024, OCR2 = 255를 사용할 때인데,

$$[0270] \quad [1024 \times (255 + 1)] / (16 \times 10^6) = 0.016384$$

[0271] 이므로, Timer 2를 이용할 경우 16.384ms(61Hz)보다 더 긴 주기의 timer interrupt를 하드웨어적으로 설정하는 것은 불가능하며, 소프트웨어적인 방법으로 이를 개선해 보는 것은 가능하다.

[0272] 다음으로 실행 순서(Execution order)에 대하여 설명한다.

[0273] Simulink에서는 블록의 실행 순서를 블록들의 연결관계에 기초해서 결정하며, 만약, A의 출력이 B의 입력에 연결되고, B의 출력이 C의 입력에 연결된다면, 그리고 모든 블록들이 다이렉트 피드 스루(direct feed-through)라면, 실행 순서는 A -> B -> C의 순서가 된다.

[0274] direct feed-through를 가지지 않는 블록들은 그 블록의 출력을 필요로 하는 다른 블록보다 그 해당 블록의 출력이 먼저 계산되지만 한다면 실행 순서상에서 어떤 곳에도 올 수 있다.

- [0275] Simulink 모델에 포함된 각 블록들이 수행되는 순서, 즉, 실행 순서를 확인하려면 Simulink의 Menu에서 Format -> Block -> Displays -> Sorted Order를 수행하면 된다.
- [0276] 도 19는 Simulink에서 실행 순서를 확인해 보기 위해 구성한 Simulink model을 나타내는 도면이다.
- [0277] 도 19에 있어서, 도시된 모델은 모두 7개의 블록을 가지고 있으며, 각각의 블록은 A에서 G까지의 라벨(label)을 가지고 있다.
- [0278] 이러한 모델에서 블록들의 실행 순서를 적어보면 다음과 같다.
- [0279] A(Source) -> F(Sink) -> B -> C -> G(Source) -> D -> E(Sink) (1-1)
- [0280] 참고로, Source block으로 분류되는 블록은 신호가 나오기만 하는 블록으로, 출력만을 가지는 블록들이 이에 해당한다.
- [0281] 또한, Sink block으로 분류되는 블록은 신호가 들어가기만 하는 블록으로, 입력만을 가지는 블록들이 이에 해당한다.
- [0282] 도 19에 나타낸 Simulink model의 경우, 다음과 같은 실행 순서를 가진다 해도 같은 수행결과를 얻을 수 있다.
- [0283] A(Source) -> G(Source) -> B -> C -> D -> E(Sink) -> F(Sink) (1-2)
- [0284] 상기에 주어진 실행 순서는 Source block의 경우 실행 순서를 정할 때 상위 순위를 갖게 하고, Sink block들은 낮은 순위를 갖도록 정한 것이다.
- [0285] 실행 순서를 정하는 방식은, Simulink와 같이 그래픽 기반의 시뮬레이션을 지원하는 소프트웨어(예를 들면, CEMTool, Labview 등) 마다 독자적인 알고리즘을 사용한다.
- [0286] 이 예에서 알 수 있는 것은, Simulink의 경우, 실행 순서를 정함에 있어서 상기한 바와 같이 Source block에 우선순위를 주고 Sink block들은 낮은 순위를 할당하는 방식을 사용하지 않는다는 것이다.
- [0287] 즉, RCP 블록들은 Send Block과 Receive Block으로 분류되며, 여기서 Receive Block은 출력만을 가지므로 Source Block에 해당되고, Send Block은 입력만을 가지므로 Sink Block에 해당된다.
- [0288] 즉, 도 20을 참조하면, 도 20은 도 19의 모델에서 Source block과 Sink block 대신 Receive Block과 Send Block을 사용해서 구성한 Simulink 모델을 나타내는 도면이다.

- [0289] 도 20에 나타낸 모델에 있어서, 실행 순서는 Simulink에 의해 다음과 같이 정해진다.
- [0290] A(Receive) → F(Send) → B → C → G(Receive) → D → E(Send) (1-3)
- [0291] 상기한 내용을 통해서 알 수 있는 중요한 사실은, Receive block과 Send block의 실행 순서가 서로 뒤섞여 있을 수 있다는 것이다.
- [0292] 즉, Simulink model을 실행하게 되면 매 샘플 시간(sample time)마다 모델에 포함된 각 블록들의 출력(output)을 한 번씩 갱신하게 되는데, Receive block일 경우에는 DAQ 장치로부터 데이터를 수신하고, Send block일 경우 DAQ 장치로 데이터를 송신하는 과정이 필요하다.
- [0293] 그런데 Simulink의 실행 순서를 따를 경우, (1-2)에 따라 한 샘플 타임(sample time)마다 수신 → 송신 → 수신 → 송신의 과정이 필요하며, 이는 구현하기도 매우 번거로울 뿐 아니라, 효율면에서도 바람직하지 않다.
- [0294] 따라서 구현을 간단하게 하고 통신 효율을 개선하기 위해서는, Receive와 Send를 일괄처리하는 것이 바람직하며, 즉, Receive를 모두 처리한 다음에 Send를 일괄 처리하는 방식이 바람직하다.
- [0295] 이를 위해서는, Receive와 Send를 일괄 처리하면서도 Simulink에서 정한 실행 순서에 위배되지 않아 올바른 결과가 나올 수 있게 하기 위한 고려가 필요하다.
- [0296] 즉, USB 통신을 수행하기 위해 PC 측에서 수행되는 어플리케이션(application)에서는 하나의 USB 통신용 객체를 생성한 후 Send 및 Receive를 수행하게 될 것이나, Send block과 Receive block들이 각각 별개의 S-Function을 통해 구현되면 USB 통신 객체를 어떤 S-Function에서 생성해야 할지에 대한 문제가 발생하게 되며, 또한, 생성된 객체에 접근해서 송신 및 수신을 하는 방법도 간단하지가 않다.
- [0297] 따라서 이러한 문제를 해결하기 위해서는, 하나의 S-Function을 통해서 Send 및 Receive 블록에 해당되는 모든 RCP 블록들을 구현해야 한다.
- [0298] 계속해서, 상기한 바와 같이 하나의 S-Function을 통해 여러 종류의 Send 및 Receive 블록을 처리하고, 각 블록들이 필요한 데이터를 DAQ 장치로부터 정확하게 수신 및 송신하도록 하기 위한 방법에 대하여 설명한다.
- [0299] 먼저, 종래의 임베디드 컨트롤러 보드(Embedded controller board)에서 제어 알고리즘을 수행할 때의 일반적인 흐름은, 센서 데이터(sensor data)의 측정, 제어 데이터의 계산, 계산된 제어 데이터를 적용하는 과정이 일정한 주기로 반복되며, 이러한 과정을 일정한 주기로 반복적으로 수행하기 위해, 일반적으로 마이크로컨트롤러의 타이머 인터럽트(timer interrupt) 기능을 활용한다.
- [0300] 그러나 본 발명의 실시예에 따른 RCP 시스템은, 제어 데이터의 계산은 PC에서 실행되고 있는 Simulink에서 담당하고, 센서 데이터의 측정과 제어 데이터의 적용은 DAQ 장치에서 이루어지는 구조이다.
- [0301] 이러한 구조의 동작 방식은 도 2에 나타낸 바와 같으며, 도 2에 나타낸 흐름도에 있어서, 음영진 부분으로 처리

된 부분이 S-Function이 관여하는 부분이다.

[0302] 또한, 주기적으로 일련의 과정을 반복적으로 수행하는 동작은 ATMEGA128의 타이머 인터럽트(timer interrupt) 기능을 활용하며, 도 2에서 굵은 화살표는 PC와 DAQ 장치 사이에 이루어지는 데이터의 교환이 있음을 나타낸다.

[0303] 그러나 매트랩(Matlab) 및 시뮬링크(Simulink)는 윈도우(Windosws)와 같은 멀티태스킹(multi-tasking) OS 상에서 실행되기 때문에, 특정 샘플 시간(sample time)에서의 제어 동작(control action)이 타이머 인터럽트(timer interrupt) 주기 내에 종료되지 않을 수도 있으므로 제어기의 실시간성에 잠재적인 문제를 가지고 있다.

[0304] 계속해서, Simulink와 S-Function의 동작에 대하여 설명한다.

[0305] 여기서, RCP 블록에 사용되는 S-Function의 동작을 이해하기 위해 Simulink가 모델을 어떻게 시뮬레이션하는지를 이해하기 위하여는, 시뮬레이션 블록(Simulink block)의 매스매틱스(mathematics)에 대한 이해가 필요하고, 이는, 예를 들면, Mathworks 사의 S-Function manual을 참조할 수 있다.

[0306] 먼저, Simulink block은 입력, 상태, 출력으로 구성되며, Simulation block의 출력은 입력, 상태, 그리고 sample time의 함수이다.

[0307] 다음의 관계식은 입력, 출력, 그리고 상태 간의 관계를 나타낸다.

[0308] $y = fo(t, x, u)$ (Output)

[0309] $xc = fd(t, x, u)$ (Derivative)

[0310] $xdk+1 = fu(t, x, u)$ (Update)

[0311] 여기서, x 는 다음과 같다.

[0312]
$$x = \begin{bmatrix} xc \\ xd \end{bmatrix}$$

[0313] Simulink block은 유형에 따라 상태(state)를 포함하지 않을 수도 있고 입력과 출력중에 한 가지만을 가질 수도 있으며, 입력만 가진 블록들은 Sink block에 해당하고, 출력만을 가진 블록은 Source block에 해당한다.

[0314] 또한, Gain block은 입력과 출력을 가지지만 상태를 가지지 않는 블록이며, State space block은 상태, 입력, 출력을 모두 가지는 블록이다.

[0315] 아울러, Simulink에서 모델의 실행은 단계별로 진행되며, 먼저, 초기화 단계(initialization phase)에서는, 다음과 같은 과정이 수행된다.

- [0316] library block들을 모델에 포함
- [0317] block의 width, data type, sample time을 전파(propagation)
- [0318] block parameter evaluation
- [0319] block execution order 결정
- [0320] 메모리 할당

- [0321] 이어서, 초기화 단계가 끝나고 나면, Simulink는 시뮬레이션 루프(simulation loop)로 진입하며, 시뮬레이션 루프에서 거치게 되는 과정을 시뮬레이션 스텝(simulation step) 이라 한다.

- [0322] 각 시뮬레이션 스텝에서, Simulink는 초기화 단계에서 결정한 실행 순서(execution order)에 따라 모델에 포함된 각 블록을 실행하게 된다.

- [0323] Simulink는 매 샘플 시간(sample time) 마다 각 블록의 상태(state), 도함수(derivative), 출력(output)을 계산하는 함수를 호출하며, 이 과정은 시뮬레이션이 종료될 때까지 계속된다.

- [0324] 다음으로, Simulink가 S-function과 상호작용하는 방식에 대하여 설명한다.

- [0325] Simulink가 S-function과 상호작용하는 방식은 2가지 관점에서 살펴볼 수 있으며, 하나는 처리 관점(process perspective)이고, 또 하나는 데이터 관점(data perspective)이다.

- [0326] 더 상세하게는, 처리 관점은 시뮬레이션의 어떤 시점에서 Simulink가 S-function을 호출하는 지에 대한 것이고, 데이터 관점은 시뮬레이션하는 동안 Simulink와 S-function이 어떻게 정보를 교환하는지에 대한 것이다.

- [0327] 먼저, 처리 관점을 통해 상호작용을 살펴보면, Simulink의 시뮬레이션 엔진(simulation engine)은 적절한 시점에 S-function에서 정의된 callback method를 호출한다.

- [0328] 즉, S-function에서 정의된 함수들은, 사용자가 호출하는 것이 아니라 Simulink 엔진이 호출하므로 callback method에 해당한다.

- [0329] 도 21을 참조하면, 도 21은 Simulink와 S-function의 상호작용을 나타내는 도면으로, 구체적으로는, Simulink가 어떤 순서로 S-function의 callback method를 호출하는지를 나타내는 도면이다.

- [0330] 도 21에 나타난 바와 같이, S-function에서 제공하는 callback method 중의 일부는 항상 호출되고, 일부는 선택적으로 호출된다.

- [0331] 이어서, 도 22를 참조하여, 데이터 관점을 통한 상호작용에 대하여 설명한다.

- [0332] 즉, 도 22를 참조하면, 도 22는 S-function 블록이 Simulink가 관리하는 다양한 데이터와 상호작용하는 방식을 나타내는 도면이다.

- [0333] 구체적으로는, S-function에는 도 21에 나타난 바와 같은 다양한 callback method가 있고, 이러한 callback method를 통해 데이터들을 변형하게 된다.
- [0334] 예를 들면, mdlOutputs라는 callback method는 블록이 가진 입력 및 상태 데이터를 이용하여 출력 데이터를 계산한다.
- [0335] 또한, S-function이 지속성 기억매체(persistent memory storage)를 필요로 할 경우 작업 벡터(work vector)를 사용할 수 있으며, 만약, S-function이 지속성 기억매체를 위해 정적(static) 또는 전역(global) 변수를 사용하게 되면, 해당 S-function을 사용하는 다수 개의 인스턴스(instance)가 그 변수들을 공유하게 된다.
- [0336] 이는, Simulink model에 같은 S-function을 사용하는 블록을 여러 개 사용할 경우 이런 일이 발생하는데, 작업 벡터(work vector)를 이용하여 하나의 S-function을 사용하는 여러 개의 인스턴스를 계속해서 추적할 수 있다.
- [0337] 여기서, 도 22를 이용하여 AvrDaqSfun이 상호작용하는 데이터를 나타내는 동시에, S-function의 method들을 같이 포함하여 개념적으로 나타내보면 도 23과 같이 나타낼 수 있다.
- [0338] 즉, 도 23을 참조하면, 도 23은 AvrDaqSfun 블록이 상호작용하는 method와 데이터를 나타내는 도면이다.
- [0339] 여기서, 본 발명의 실시예에 따른 RCP에서 사용하는 S-function 블록의 경우는 상태(state)를 포함하지 않기 때문에, 상태 데이터(state data)가 없는 것에 유념해야 한다.
- [0340] 또한, 파라미터 데이터(parameter data)는 Simulink를 통해서 제공받는 것으로서, 파라미터 값(parameter value)의 변경은 GUI 방식의 대화상자를 통해서 이루어진다.
- [0341] 아울러, 처리 관점의 측면에서 AvrDaqSfun을 살펴보면, 도 24의 흐름도와 같이 나타낼 수 있다.
- [0342] 즉, 도 24를 참조하면, 도 24는 도 2의 흐름도를 S-function의 callback method를 통해 나타내고 있으며, 더불어 같이 통신하면서 수행되는 DAQ 장치의 동작 흐름도를 좀 더 구체화하여 나타내고 있다.
- [0343] 도 24에 나타난 바와 같이, AvrDaqSfun을 이용한 RCP 블록들은 상태를 가지지 않으며 입력과 출력중의 하나만을 가지는 블록으로 S-function에서 제공하는 다양한 callback method들 중에서 mdlInitializeSizes, mdlInitializeSampleTimes, mdlStart, mdlOutputs, mdlTerminate의 5개의 callback method만으로 구현된다.
- [0344] 또한, 도 24에 있어서, 붉은색으로 표시된 화살표는 USB 통신을 통한 데이터의 흐름을 나타낸다.
- [0345] 다음으로, 도 25를 참조하면, 도 25는 callback method 중에서 mdlInitializeSizes와 mdlStart의 처리를 나타내는 플로차트이고, 도 26은 callback method 내에서 호출되는 주요 함수인 InitCommunication()의 처리를 나타내는 플로차트이며, 도 27은 callback method인 mdlOutputs()의 처리를 나타내는 플로차트이다.

- [0346] 계속해서 상기한 도 25 내지 도 27의 흐름도를 참조하여, 각 함수들의 기능에 대하여 설명한다.
- [0347] 여기서, 설명의 편의를 돕기 위해, 도 12에 나타낸 Simulink model을 예로 들어 설명하기로 한다.
- [0348] 여기서, 도 12에 나타낸 모델에 포함된 블록들에 병기된 숫자는 실행 순서를 나타내며, 따라서 이 모델에서는 ADC 블록이 실행 순서가 가장 앞서는 블록임을 알 수 있다.
- [0349] 즉, 도 12에 나타낸 Simulink model에 포함된 10개의 RCP 블록들의 실행순서를 정리해 보면 다음과 같다.
- [0350] ADC -> Buzzer -> DAC -> DI -> DO -> ENC -> PWM -> LED -> Time -> Switch
- [0351] 여기서, 실행순서는 RCP 블록들만을 대상으로 했을 때의 실행순서를 나타낸다.
- [0352] **mdlCheckParameters**
- [0353] 각각의 RCP 블록은 BlockType, SampleTimeInfo, Re-ceiveDataInfo, SendDataInfo의 모두 4개의 파라미터를 넘겨받고, 각각의 파라미터가 의미 있는 값을 가지는지 검사한다.
- [0354] 예를 들면, 인코더 블록(Encoder block)은 여러 개의 채널을 가지므로, 사용자가 적어도 1개 이상의 채널을 사용하도록 설정했는지 검사하고, 만약 그렇지 않다면 오류를 발생시킨다.
- [0355] 또한, 함수 mdlCheck-Parameters는 함수 mdlInitializeSizes 안에서 호출된다.
- [0356] **mdlInitializeSizes**
- [0357] 먼저, mdlCheckParameters를 호출하여 파라미터의 적합성 여부를 검사하고(도 25의 A1), RCP block이 넘겨받는 파라미터인 ReceiveDataInfo 혹은 SendDataInfo로부터 블록이 몇 개의 출력(Receive block의 경우) 포트 또는 입력 포트(Send block의 경우)를 가져야 하는지를 설정한다(도 25의 A3 ~ A8).
- [0358] 다음으로, AvrDaqSfun.cpp에 사용되고 있는 전역변수를 0으로 초기화하고(도 25의 A9), 초기화하는 작업이 필요 없을 경우는 처음 실행했을 때에만 변수값이 0이고 두 번째 이후의 실행에는 이전 실행에서 남아 있던 값이 사용되게 된다.
- [0359] 따라서 매번 실행할 때마다 같은 결과를 얻기 위해서 mdlInitializeSizes 라는 함수의 이름과 맞지는 않지만 전역변수들의 초기화 작업을 여기서 해준다.
- [0360] **mdlInitializeSampleTimes**
- [0361] RCP 블록의 샘플 시간 정보를 설정하며, 본 실시예에서 RCP 블록은 루트(root)에서 설정된 샘플링 시간(sampling time)을 사용한다.

- [0362] 따라서 RCP 블록들이 모두 같은 샘플 시간을 사용하도록 설정된다.
- [0363] **mdlStart**
- [0364] 객체지향 프로그래밍의 개념으로 설명하면, RCP 블록은 execution order(변수명:ExecOrder), receive data offset(변수명:RxDataOffset), send data offset(변수명:TxDataOffset)과 같은 멤버 데이터(member data)를 가진다.
- [0365] S-Function 상에서 이러한 멤버 데이터는 작업 벡터(Work vector)를 이용하여 구현한다(도 23 참조).
- [0366] 각각의 RCP 블록이 가지고 있는 이 3개의 멤버 데이터는 Simulation loop가 시작되기 전에 파악되어야 하는데, mdlStart가 수행하는 역할중 하나는 각각의 RCP 블록에 대하여 이 3개의 멤버 데이터를 알아내어 갱신하는 것이다.
- [0367] 도 25에 있어서, B1은 각 RCP 블록의 ExecOrder를 갱신하는 역할을, B2 ~ B5는 TxDataOffset을 갱신하는 역할을, B6 ~ B8은 RxDataOffset을 갱신하는 역할을 수행한다.
- [0368] 여기서, ExecOrder는 특정 Simulink model에 포함된 RCP 블록들 사이에서의 시퀀스 순서를 나타내며, 가장 순서가 빠른 경우가 1이다.
- [0369] 본 발명의 실시예에 따른 RCP 시스템은 상기한 바와 같이 Simulink model 내에 포함된 RCP 블록과 관련된 데이터를 일괄적으로 송신/수신하며, 이러한 일괄 송신/수신 방식은 통신효율을 개선함으로써 제어 시스템의 샘플 시간을 작게 설정할 수 있도록 하기 위함이다.
- [0370] 도 12에 나타난 Simulink model에 대하여, 본 발명의 실시예에 따른 RCP 시스템에서는 수신 데이터 프레임을 도 28과 같이 구성하고 송신 데이터 프레임은 도 29와 같이 구성하여 일괄 송신/수신을 수행한다.
- [0371] 즉, 도 28을 참조하면, 도 28은 본 발명의 실시예에 따른 RCP 시스템의 수신 데이터 프레임의 구성을 나타내는 도면이고, 도 29는 본 발명의 실시예에 따른 RCP 시스템의 수신 데이터 프레임의 구성을 나타내는 도면이다.
- [0372] 각각의 데이터 프레임 상에서 특정 RCP 블록과 관련된 데이터가 나타나는 순서는 ExecOrder를 따른다.
- [0373] TxDataOffset은 송신 데이터 프레임 상에서 특정 RCP 블록과 관련된 데이터가 시작되는 지점을 나타내며, RxDataOffset은 수신 데이터 프레임 상에서 특정 RCP 블록과 관련된 데이터가 시작되는 지점을 나타낸다.
- [0374] USB 통신 수행시, 데이터는 ASCII encoding을 사용하며, 도 28과 도 29에 나타난 데이터의 byte 수는 ASCII encoding을 사용하였을 때의 byte 수이다.
- [0375] 도 12에 나타난 Simulink model의 모든 블록 대하여 mdlStart가 완료되면, 각 RCP 블록에 대해 도 30의 표에

나타낸 바와 같이 멤버 데이터 갱신이 완료된다.

- [0376] 즉, 도 30을 참조하면, 도 30은 도 12에 나타낸 Simulink 모델에 사용된 RCP 블록들의 멤버 데이터를 표로 정리하여 나타내는 도면이다.
- [0377] 여기서, mdlStart의 또 다른 중요 기능은, 수신 데이터 및 송신 데이터의 정보를 담고 있는 배열(array)을 완성하는 것이다(도 25의 B5, B9 참조).
- [0378] RxDataInfo는 PC가 DAQ 장치로부터 수신해야 할 데이터의 종류와 순서 정보를 담고 있으며, TxDataInfo는 PC측에서 DAQ 장치로 전송해야 하는 데이터의 종류와 순서를 담고 있다.
- [0379] 이러한 정보를 Simulation loop가 시작되기 전에 DAQ 장치에 송신함으로써 DAQ 장치가 측정하여 PC 쪽으로 송신 해주어야 하는 데이터의 종류와 순서를 지정해 줄 수 있고, DAQ 장치가 PC로부터 어떤 종류의 신호가 어떤 순서로 전송되어 오는지를 알려 줄 수 있게 된다.
- [0380] 또한, TxDataInfo와 RxDataInfo는 각 RCP 블록의 실행 순서에 따라 순서대로 채워지게 된다.
- [0381] 구체적으로는, 도 31을 참조하면, 도 31은 도 12의 Simulink 모델에 대해 각각의 RCP 블록에 대한 mdlStart가 모두 완료되면 생성되는 TxDataInfo와 RxDataInfo를 개념적으로 나타낸 것이다.
- [0382] mdlStart를 끝내게 되면, 송신해야 할 데이터의 byte 수와 수신해야 할 데이터의 byte 수가 각각 변수 BytesOfTxData와 BytesOfRxData에 남게 되고(도 25의 B4, B8참조), 이 정보는 mdlOutputs에서 일괄 송신/수신을 수행할 때 몇 byte의 데이터를 송/수신해야 하는지를 결정할 때 사용된다.
- [0383] 다음으로, 하나의 Simulink model에 모두 몇 개의 RCP 블록이 사용되었는지를 알아내고, mdlStart 가 모두 완료되면 B1에서 갱신되는 변수 NumOfBlock에 RCP 블록의 수가 남게 된다.
- [0384] 여기서, 도면에 도시되는 않았으나, 특정 RCP 블록이 하나의 Simulink model에서 2번 이상 사용되었는지를 검사하고, 만약 그렇다면 오류메시지를 출력하도록 구성될 수 있다.
- [0385] 예를 들면, PWM 블록이 2개가 사용되었다면, 이것을 지적하는 오류메시지를 출력한다.
- [0386] **mdlOutputs**
- [0387] 도 24에 나타낸 바와 같이, Simulation loop로 진입하게 되면 매 샘플 시간마다 Simulink에 의해 각 RCP 블록의 callback method 중에서 mdlOutputs가 호출된다.
- [0388] RCP block은 연속형 상태를 포함하고 있지 않은 블록이므로, mdlOutputs는 메이저 타임 스텝(Major Time Step)에서만 실행되며, 이는, 제어 알고리즘의 경우, 하나의 샘플 동안 1회의 제어 연산만 수행하기 때문이다.

- [0389] 여기서, 메이저 타임 스텝(Major time step)과 마이너 타임 스텝(Minor time step)의 차이에 대해서는 도 21을 참조할 수 있다.
- [0390] 또한, mdlOutputs는 크게 5가지의 조건을 순서대로 점검하여 각 조건이 만족할 경우 해당하는 작업을 수행하며, 즉, 도 26에 나타난 mdlOutputs의 흐름도에서, 5가지 조건을 검사하는 부분은 각각 C1, C3, C5, C7, C9이다.
- [0391] 여기서, 앞서 설명한 바와 같이, 이러한 5가지의 조건은 서로 배타적이 아니므로 2개 이상의 조건이 만족되어 2개 이상의 작업이 수행될 수도 있다.
- [0392] 더 상세하게는, 먼저, $t = 0$ 인 시점이고, 실행 순서가 가장 앞서는 RCP 블록인 경우(도 26의 C1)(여기서, 각 RCP 블록의 실행 순서는 mdlStart의 수행이 모두 완료되면 파악되며, 실행 순서는 RCP 블록들 사이에서의 실행 순서를 말한다.), 함수 InitCommunication()을 호출한다(도 26의 C2).
- [0393] 즉, 도 27을 참조하면, 도 27은 함수 InitCommunication()의 흐름도를 나타내는 도면으로, 도 27에 나타난 바와 같이, 함수 InitCommunication()에서는 USB 통신을 위한 통신객체를 생성하고(도 27의 D1), DAQ 장치와 USB 통신을 연결하는 작업을 수행한다(도 27의 D2, D3).
- [0394] 그리고 DAQ 장치에서 필요한 여러 가지 정보(SampleTimeInfo, TxDataInfo, RxDataInfo)를 전달한다(도 27의 D4).
- [0395] 여기서, mdlOutputs는 Simulation loop에 진입했을 때 반복수행되는 함수이므로, 1회성으로 끝나는 작업들은 개념상으로 mdlOutputs에서 수행되는 것이 아니라 mdlStart에서 수행되는 것이 맞지만, DAQ 장치 측으로 전달해 주어야 하는 정보들 중 일부가 mdlStart의 수행이 끝나야만 모두 얻어지기 때문에, 1회성 작업이지만 mdlOutputs에서 수행한다.
- [0396] 단, $t = 0$ 이고 실행 순서가 가장 앞서야 한다는 조건으로 인해, 이 경우에 해당하는 작업은 반복적으로 수행되지 않고 1회만 수행된다.
- [0397] 또한, 실행 순서가 가장 앞서는 RCP 블록일 경우(도 26의 C3), DAQ 장치로부터 일괄적으로 센서 데이터를 수신하여 배열(변수명: RxBuffer)에 채워 넣는다(도 26의 C4).
- [0398] 여기서, 도면상에 도시되어 있지는 않으나, 일괄 수신한 데이터의 byte 수가 실제 받기로 되어 있는 데이터의 byte 수와 같은 지를 확인하고, 만약 다르다면 오류메시지 발생과 함께 빠져나오는 처리도 C4 단계에 포함된다.
- [0399] 다음으로, RCP 블록의 타입(type)이 Receive block인 경우는(도 26의 C5), C3, C4 단계를 통해 DAQ 장치로부터 센서 데이터를 일괄 수신하여 RxBuffer가 채워진 상태이다.
- [0400] 따라서 RxBuffer로부터 해당 RCP 블록에서 필요한 데이터만을 추출해 블록의 출력을 갱신하며, 이때 오프셋(offset) 정보와 해당 RCP 블록이 만약 채널을 가진 블록이라면 채널 정보를 이용한다.

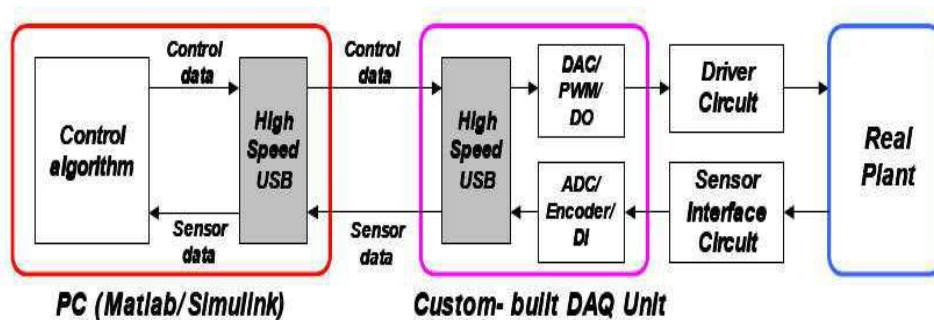
- [0401] 여기서, 채널의 개수는 Simulink로부터 넘겨받은 ReceiveDataInfo의 사이즈(size) 정보로부터 얻어낼 수 있다.
- [0402] 더 상세하게는, 도 12의 Simulink model에서 인코더 블록(Encoder block)을 예로 들어 설명하면, 인코더 블록의 파라미터인 ReceiveDataInfo가,
- [0403] `ReceiveDataInfo = [0x10;0x12]`
- [0404] 이므로, 2개의 채널을 가지는 것을 알 수 있으며, 또한, 도 30에 나타난 바와 같이, 인코더 블록의 멤버 데이터 중에서 RxDataOffset = 10이므로, 도 28을 참조하면, 배열 RxBuffer의 10번째 원소(element)부터 16 byte의 데이터가 인코더 블록에 해당하는 데이터임을 알 수 있다.
- [0405] 따라서 RxBuffer로부터 인코더 블록에 해당하는 데이터를 추출하여 인코더 블록의 출력을 갱신하면 된다.
- [0406] 계속해서, RCP의 타입이 Send block인 경우(도 26의 C7), 오프셋 정보와 채널 수 정보를 이용하여 배열 TxBuffer를 채워 넣는다.
- [0407] 도 12의 Simulink model에서 PWM 블록을 예로 들어 설명하면, PWM 블록의 파라미터인 SendDataInfo가
- [0408] `SendDataInfo = [0x80;0x82]`
- [0409] 이므로, 2개의 입력을 가지는 것을 알 수 있으며, 또한, 도 30을 참조하면, PWM 블록의 멤버 데이터 중에서 TxDataOffset = 12이므로, 도 29를 참조하여, PWM 블록의 입력 데이터 2개를 배열 TxBuffer의 12번째 원소부터 채워넣으면 된다.
- [0410] 상기한 바와 같이 하여, 도 12의 Simulink model에 포함된 5개의 Send block에 대해 mdlOutputs가 모두 호출되면, TxBuffer에는 5개의 Send block에 대한 데이터가 채워지게 되며, 이렇게 채워진 데이터는 나중에 DAQ 장치 쪽으로 일괄 전송된다.
- [0411] 다음으로, 실행 순서가 가장 늦은 RCP 블록일 경우(도 26의 C9), TxBuffer에 채워진 데이터를 DAQ 장치 쪽으로 일괄 송신한다(도 26의 C10).
- [0412] 참고로 TxBuffer에 채워진 데이터의 byte 수는 BytesOfTxData라는 변수에 저장되어 있다.
- [0413] **mdlTerminate**
- [0414] 실행 순서가 가장 늦은 RCP 블록에서 'z'를 DAQ 장치로 송신하며, DAQ 장치는, 'z'를 수신할 경우, 지금 현재 세션(session)이 종료되고 또 다른 세션이 시작되는 것을 대비하여 대기 상태가 된다.
- [0415] 또한, USB 통신을 마치고 전에 수신 종료점(Endpoint)에 남아 있던 데이터를 불러 읽어들이며, 또한, 송신 및 수신 종료점을 리셋(reset) 시킨다.

- [0416] 아울러, USB device를 접근하기 위해 생성했던 객체(object)를 소멸시킨다.
- [0417] 따라서 상기한 바와 같은 내용에 근거하여, 본 발명에 따른 Matlab과 DAQ 장치를 이용한 RCP 시스템을 구현할 수 있다.
- [0418] 즉, 본 발명에 따른 RCP 시스템은, 상기한 바와 같이, 기존의 RCP 시스템 및 대학교에서 사용하는 학부 또는 대학원용 실험실습 기자재들이 대부분 자체개발이 아니라 외부에서 구매한 것이어서 유지, 보수에 어려움을 겪는 점을 고려해 보면, 본 발명의 실시예에 따른 RCP 시스템은 자체 개발된 것이므로 유지, 보수의 편리성 면에서 큰 장점을 가진다.
- [0419] 또한, 본 발명에 따른 RCP 시스템은, 비교적 저가의 마이크로컨트롤러(ATMEGA128)를 이용하고, 아울러, Realtime Workshop이나 Embedded coder와 같이 기존의 RCP 시스템에서 필요로 하는 고가의 소프트웨어가 필요 없이 가장 기본적인 Matlab/Simulink 만으로 구성 가능하므로, 많은 대학교에서 실험 실습용으로 기본적인 Matlab/Simulink를 가지고 있는 점을 감안하면, 본 발명의 실시예에 따른 RCP 시스템을 매우 용이하게 적용할 수 있다.
- [0420] 더욱이, 본 발명에 따른 RCP 시스템은, 기존의 RCP 시스템과는 달리 Simulink 자체가 제어기 기능을 수행하고 C-code를 생성하는 과정과 컴파일의 필요 없으므로, 제어를 변경하고 실험해 보는 데 소요되는 시간이 크게 단축될 수 있으며, 또한, DAQ 장치가 USB 통신을 사용하여 장치가 특정 PC에 종속될 필요가 없으므로, 학교에서 뿐만 아니라 집에서 실험이 가능하여 학생들의 학습 효율향상에 크게 기여할 수 있다.
- [0421] 이상, 상기한 바와 같은 본 발명의 실시예를 통하여 본 발명에 따른 Matlab과 DAQ 장치를 이용한 RCP 시스템의 상세한 내용에 대하여 설명하였으나, 본 발명은 상기한 실시예에 기재된 내용으로만 한정되는 것은 아니며, 따라서 본 발명은, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에 의해 설계상의 필요 및 기타 다양한 요인에 따라 여러 가지 수정, 변경, 결합 및 대체 등이 가능한 것임은 당연한 일이라 하겠다.
- [0422] [참고문헌]
- [0423] 1. The Mathworks Inc., Real-time workshop user's guide(rtw ug.pdf), Version 6, 2005.
- [0424] 2. The Mathworks Inc., Real-time workshop embedded code user's guide (ecoder ug.pdf), Version 4, 2005.
- [0425] 3. S. Rebeschies, "MIRCOS-Microcontroller-based real time control system toolbox for use with Matlab/Simulink," in Proc. IEEE Int. Symp. Computer Aided Control System Design, 1999, pp. 267-272.
- [0426] 4. K. H. Hong, W. S. Gan, Y. K. Chong, K. K. Chew, C. M. Lee, and T. Y. Koh, "An integrated environment for rapid prototyping of DSP algorithms using and Texas Instruments' TMS320C30," Microprocessors and Microsystems, vol. 24, no. 7, pp. 349-363, 2000.
- [0427] 5. W. Lee, M. Shin, and M. Sunwoo, "Target-identical rapid control prototyping platform for model-based engine control," Proc. InstnMech. Engrs Part D, J. Automobile Engineering, vol. 218, pp. 755-765, 2004.

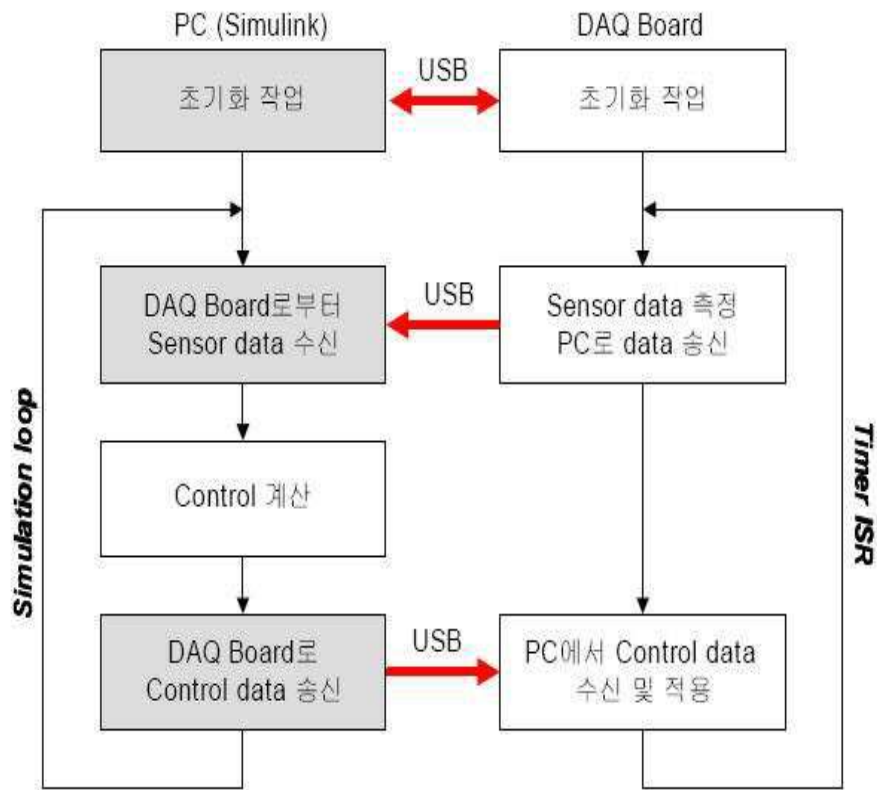
- [0428] 6. D. Hercog and K. Jezernik, "Rapid control prototyping using Matlab/Simulink and a DSP-based motor controller," *Int. J. Engng Ed.*, vol. 21, no. 3, pp. 1-9, 2005.
- [0429] 7. R. Bucher and S. Balemi, "Rapid controller prototyping with Matlab/Simulink and Linux," *Control Engineering Practice*, vol. 14, pp. 185-192, 2006.
- [0430] 8. M. I. Khalil, "A USB-based data acquisition system for neutron TOF measurements," *Meas. Sci. Technol.*, vol. 17, pp. N1-N7, 2006.
- [0431] 9. V. Fanti, R. Marzeddu, G. Piredda, and P. Randaccio, "A portable acquisition system based on USB standard for the Medipix2 X-ray detector," *IEEE Trans. Nuclear Science*, vol. 53, no. 5, pp. 2578-2583, 2006.
- [0432] 10. H. Jiang, M. Ojaruega, F. D. Becchetti, H. C. Griffin, and R. O. Torres-Isea, "A USB-based portable data acquisition system for detector development and nuclear research," *Nuclear Instruments and Methods in Physics Research A*, vol. 652, pp. 483-486, 2011.
- [0433] 11. "Universal Serial Bus Specification," www.usb.org.

도면

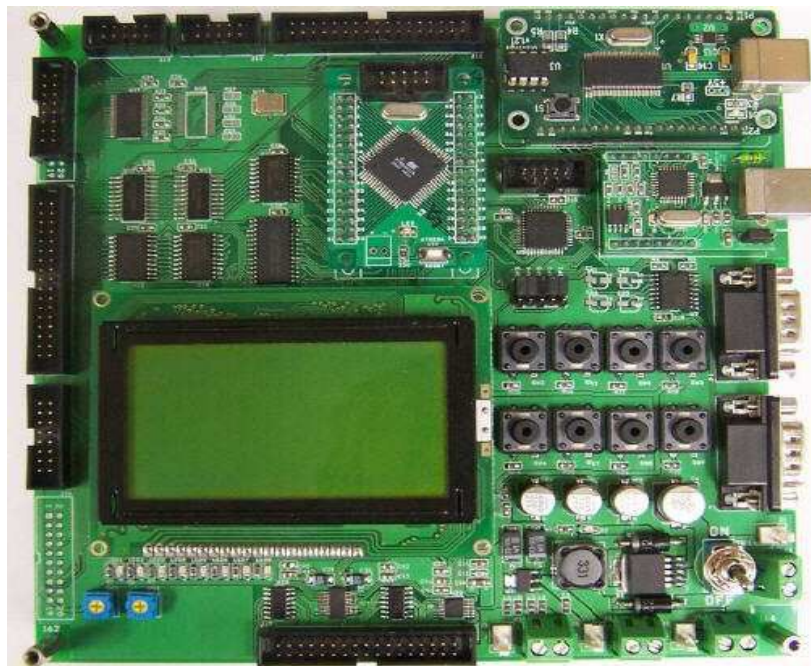
도면1



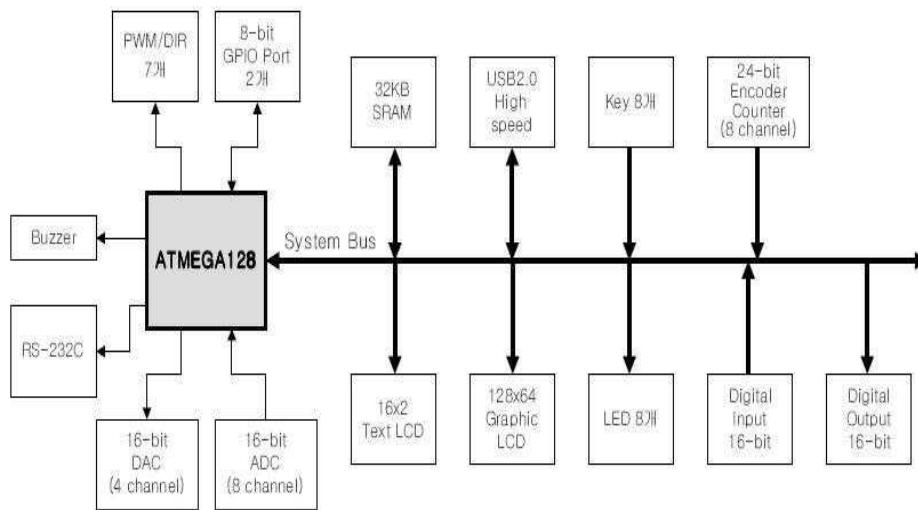
도면2



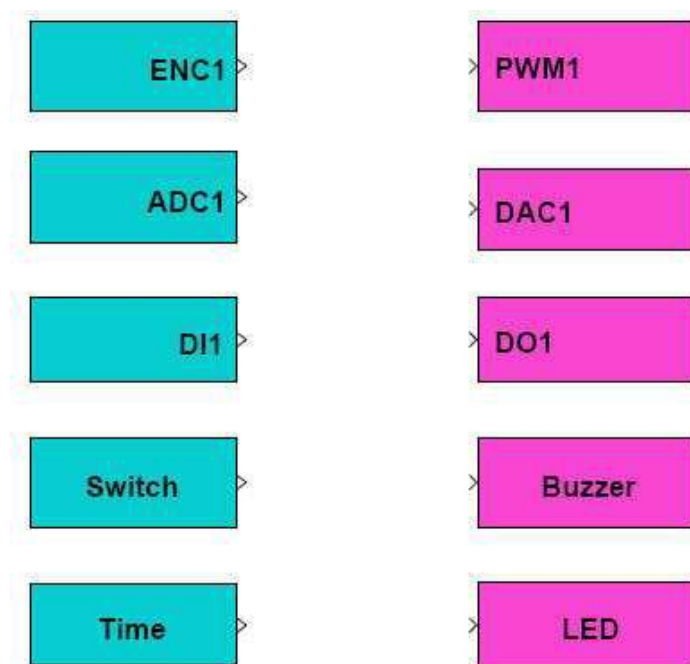
도면3



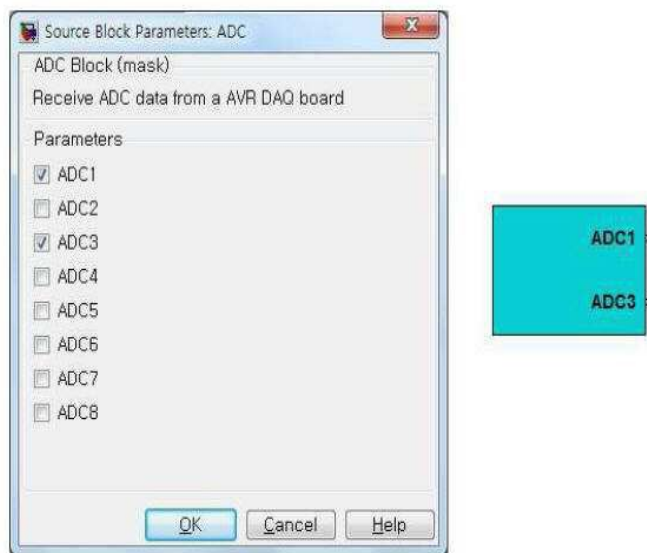
도면4



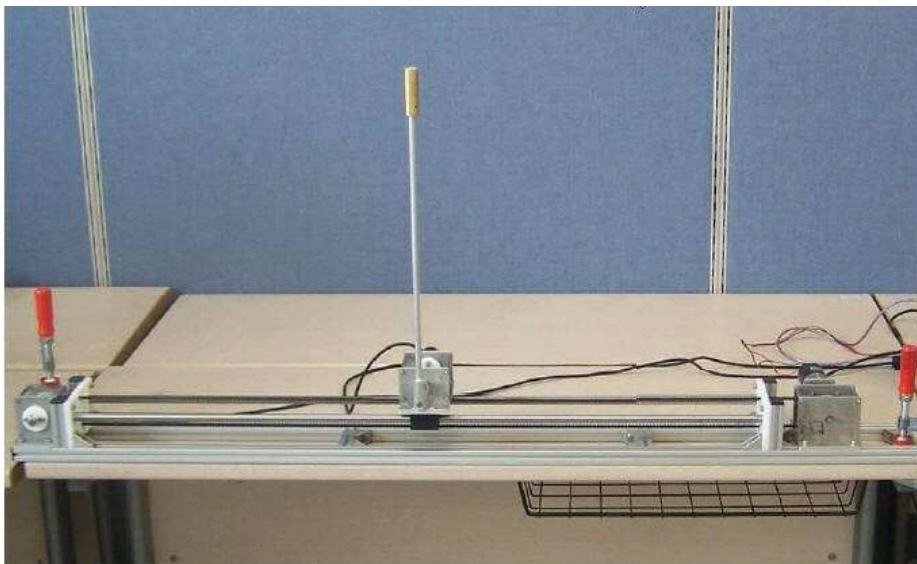
도면5



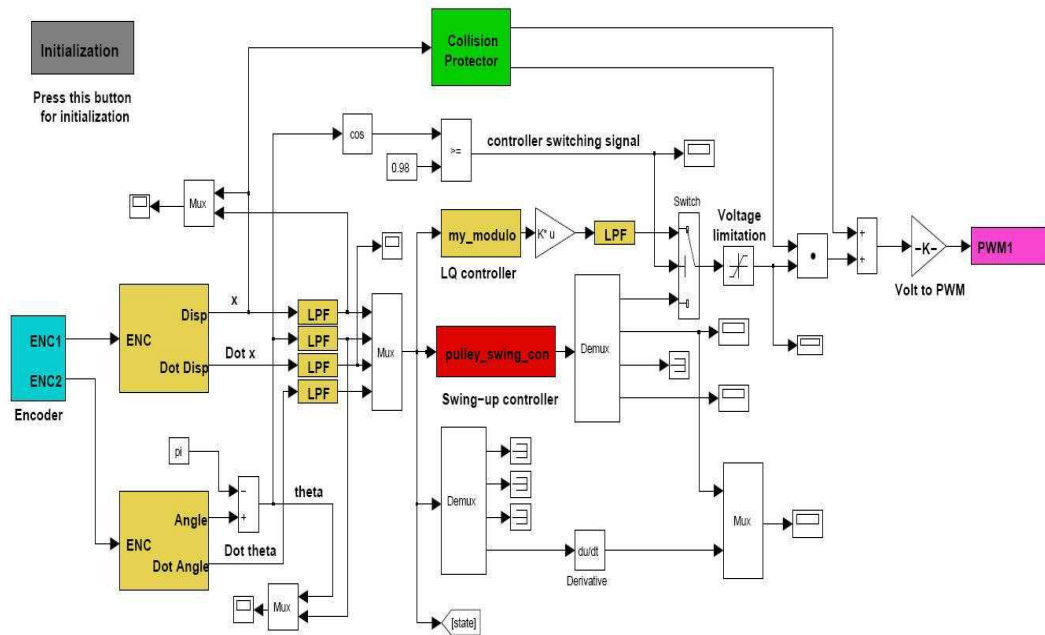
도면6



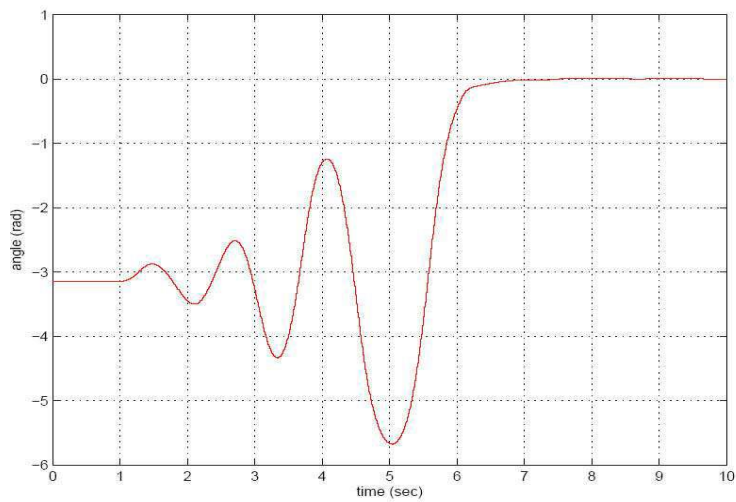
도면7



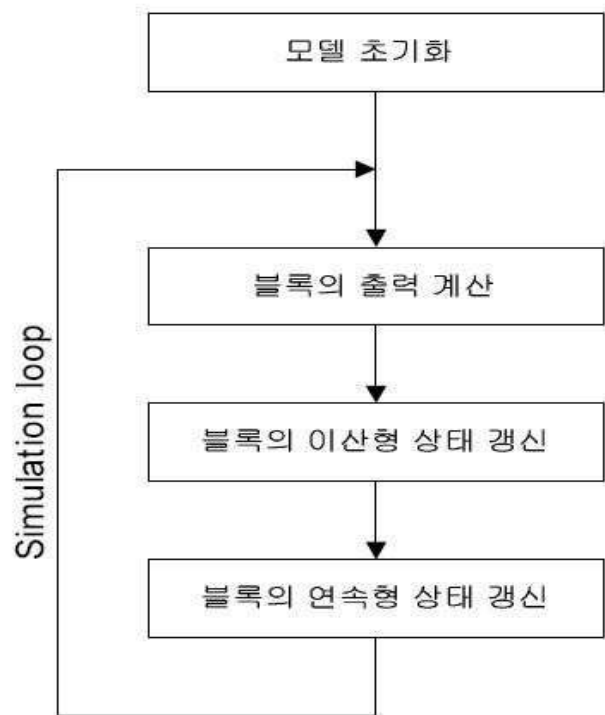
도면8



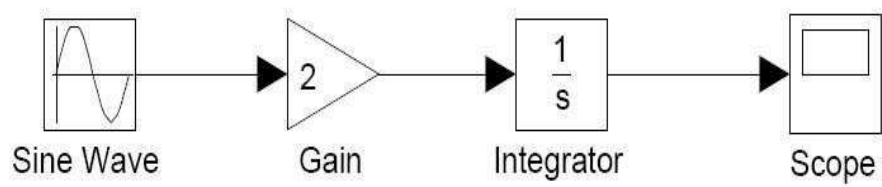
도면9



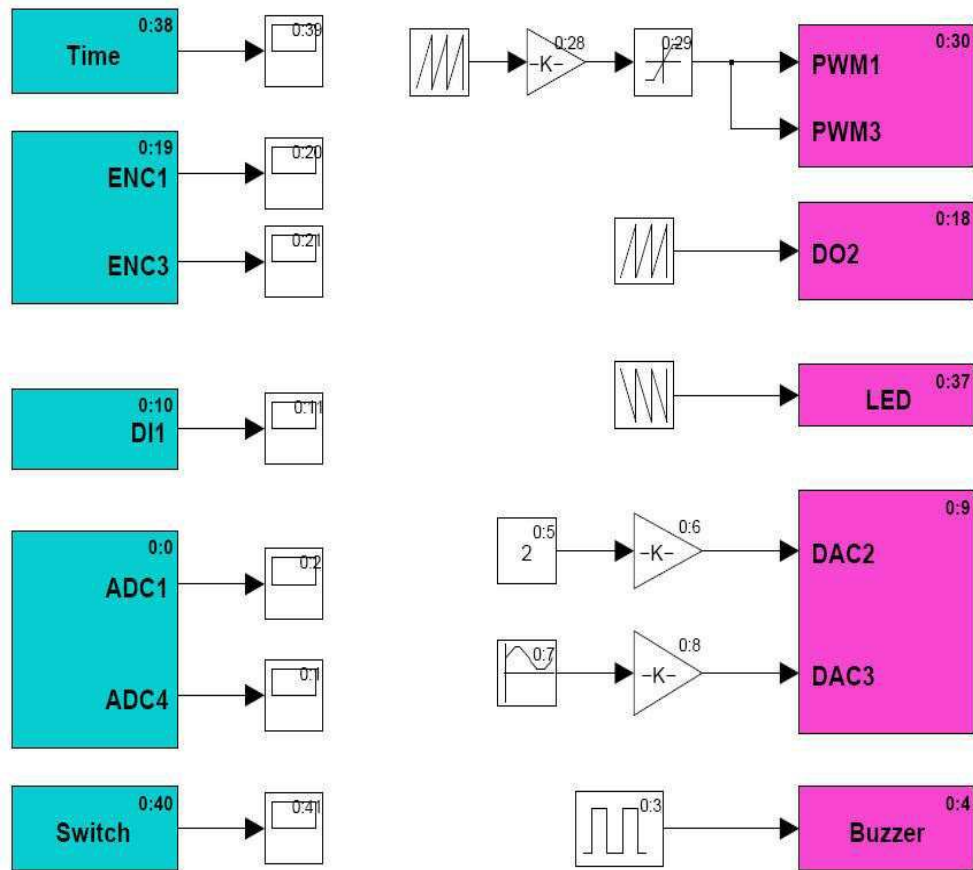
도면10



도면11



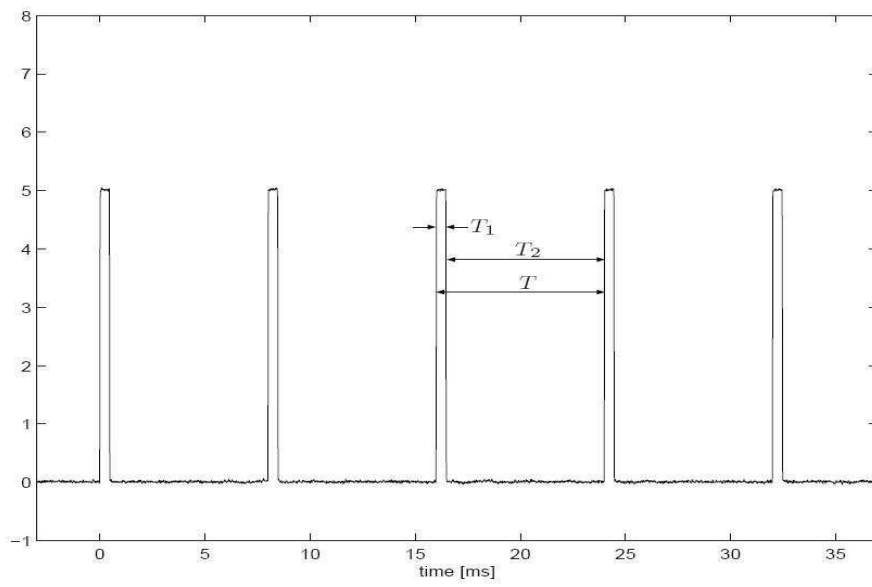
도면12



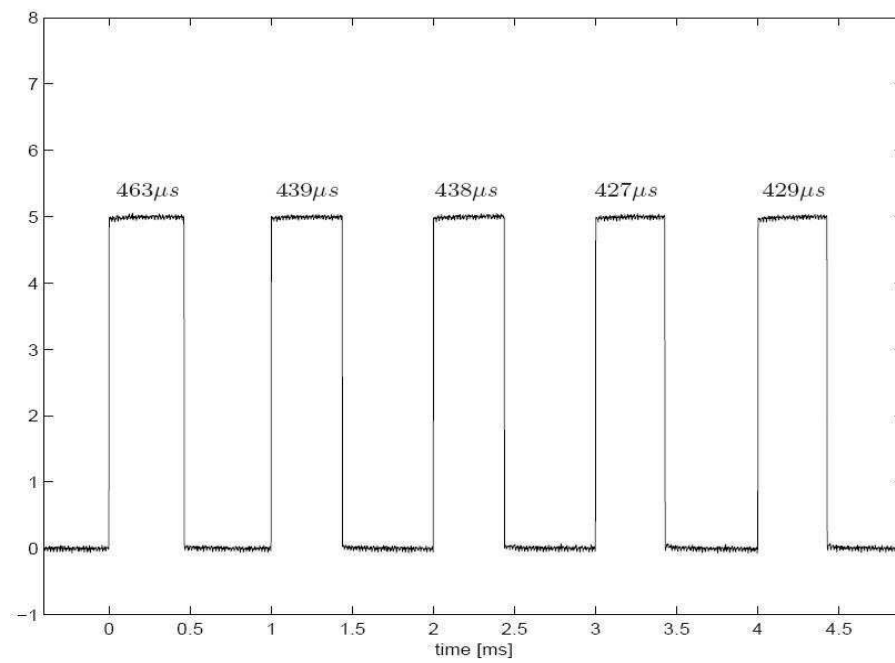
도면13

RxDataInfo	TxDataInfo
ADC CH1	Buzzer
ADC CH4	DAC CH2
Digitla Input CH1	DAC CH3
Encoder CH1	Digital Output CH2
Encoder CH3	PWM CH1
Time	PWM CH3
Switch	LED

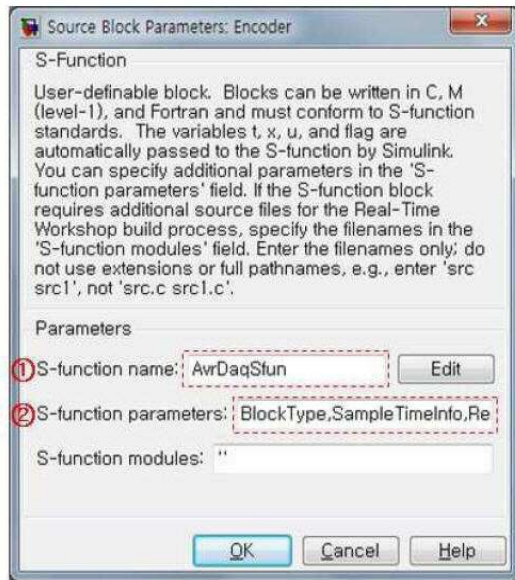
도면14



도면15



도면16



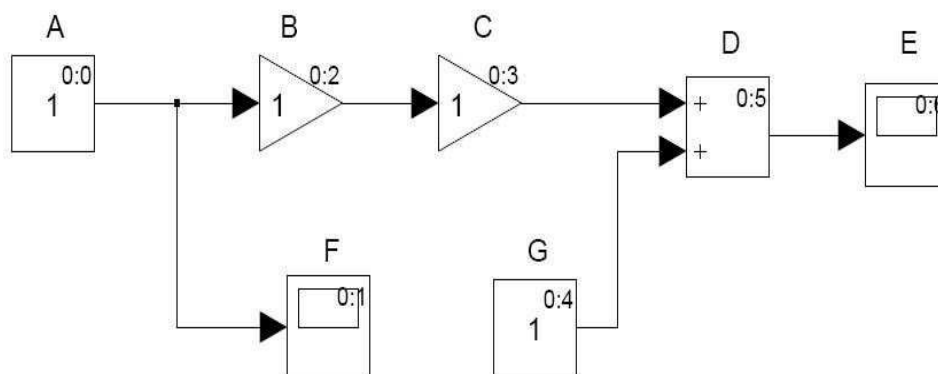
도면17

	Block type	Block Type Number
Receive Block	Time	0x00
	Encoder	0x10
	ADC	0x20
	Digital Input	0x30
	Switch	0x40
Send Block	PWM	0x80
	DAC	0x90
	LED	0xA0
	Digital Output	0xB0
	Buzzer	0xC0

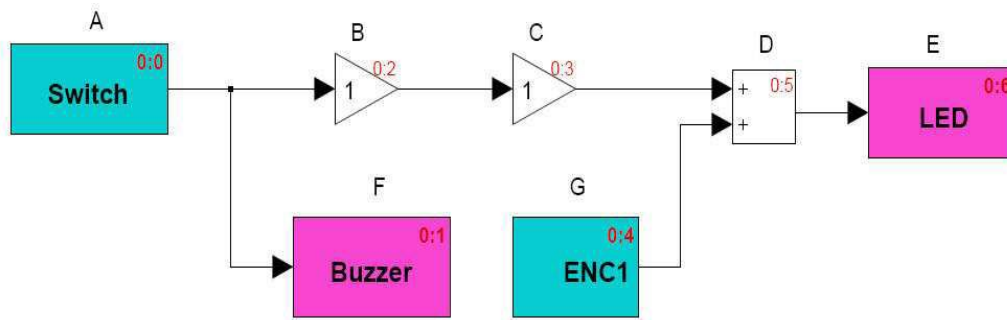
도면18

Block type	Channel	ReceiveDataInfo	Block type	Channel	SendDataInfo
Time	N/A	0x00	PWM	1	0x80
Encoder	1	0x10		2	0x81
	2	0x11		3	0x82
	3	0x12		4	0x83
	4	0x13		5	0x84
	5	0x14		6	0x85
	6	0x15		7	0x86
	7	0x16	DAC	1	0x90
	8	0x17		2	0x91
ADC	1	0x20		3	0x92
	2	0x21		4	0x93
	3	0x22	LED	N/A	0xA0
	4	0x23	Digital Output	1	0xB0
	5	0x24		2	0xB1
	6	0x25	Buzzer	N/A	0xC0
	7	0x26			
	8	0x27			
Digital Input	1	0x30			
	2	0x31			
Switch	N/A	0x40			

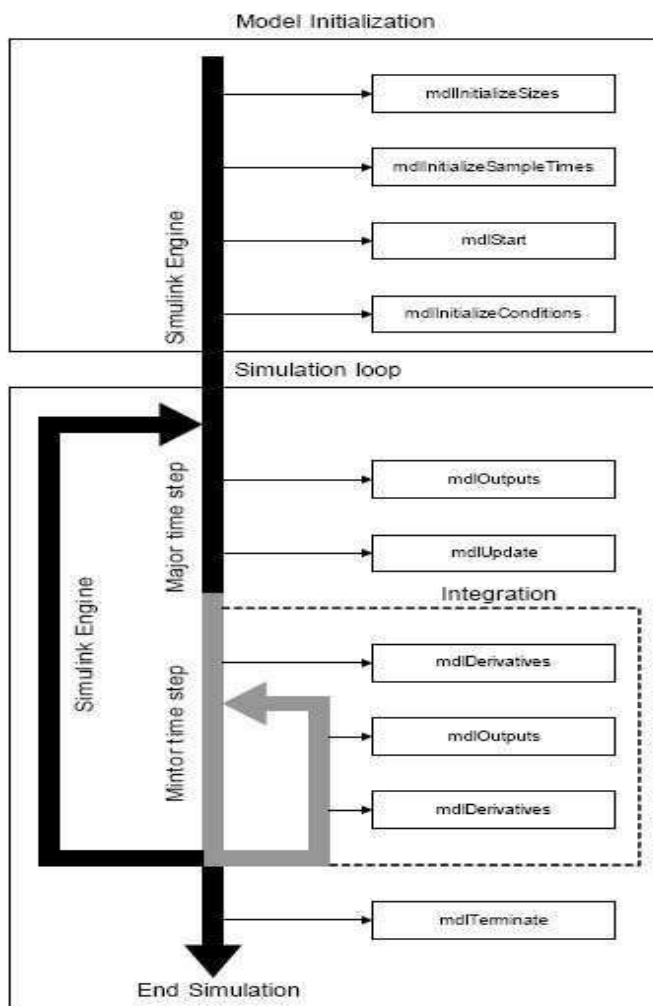
도면19



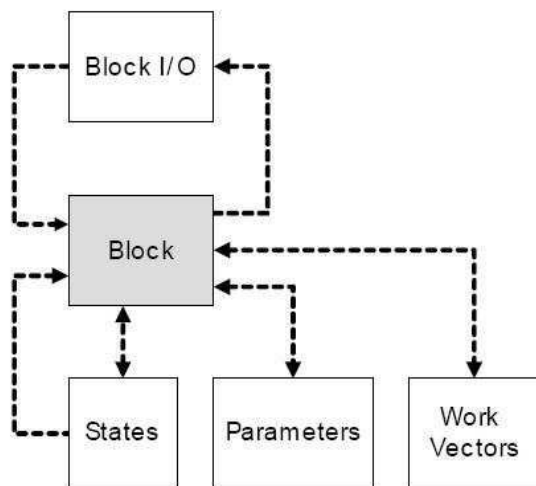
도면20



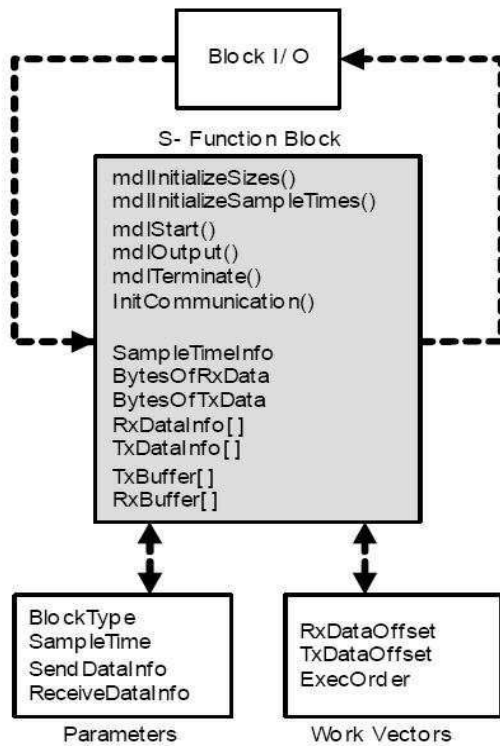
도면21



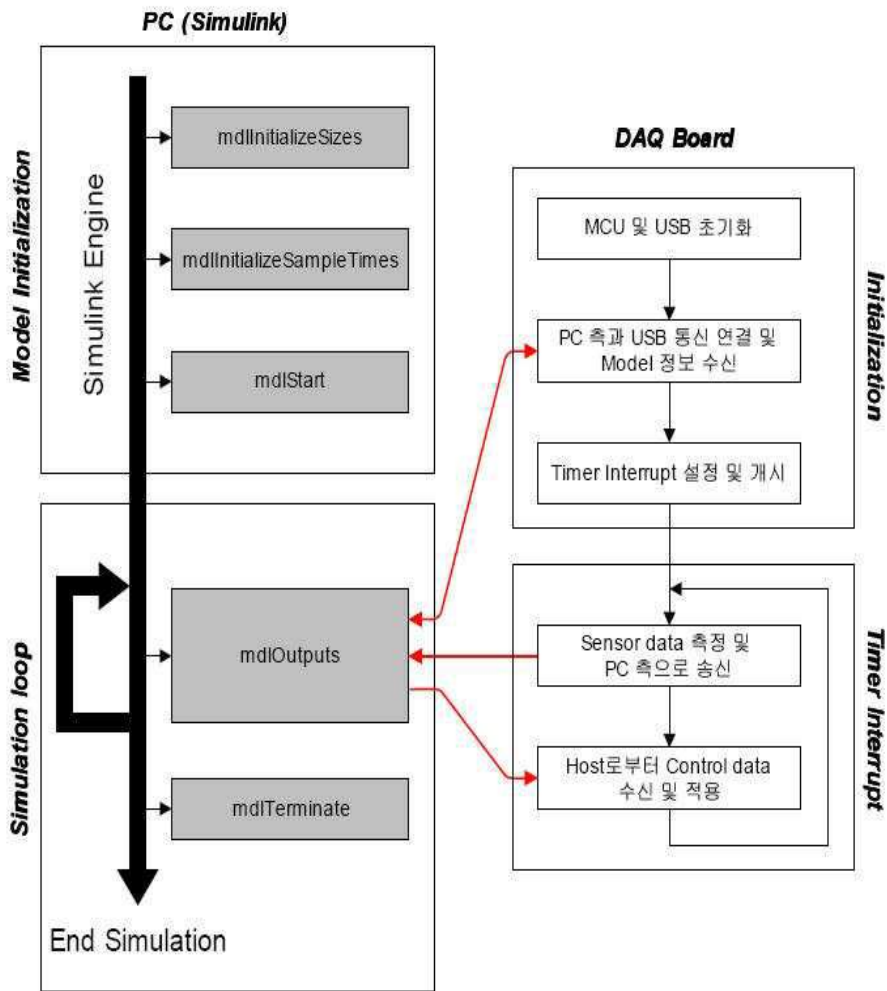
도면22



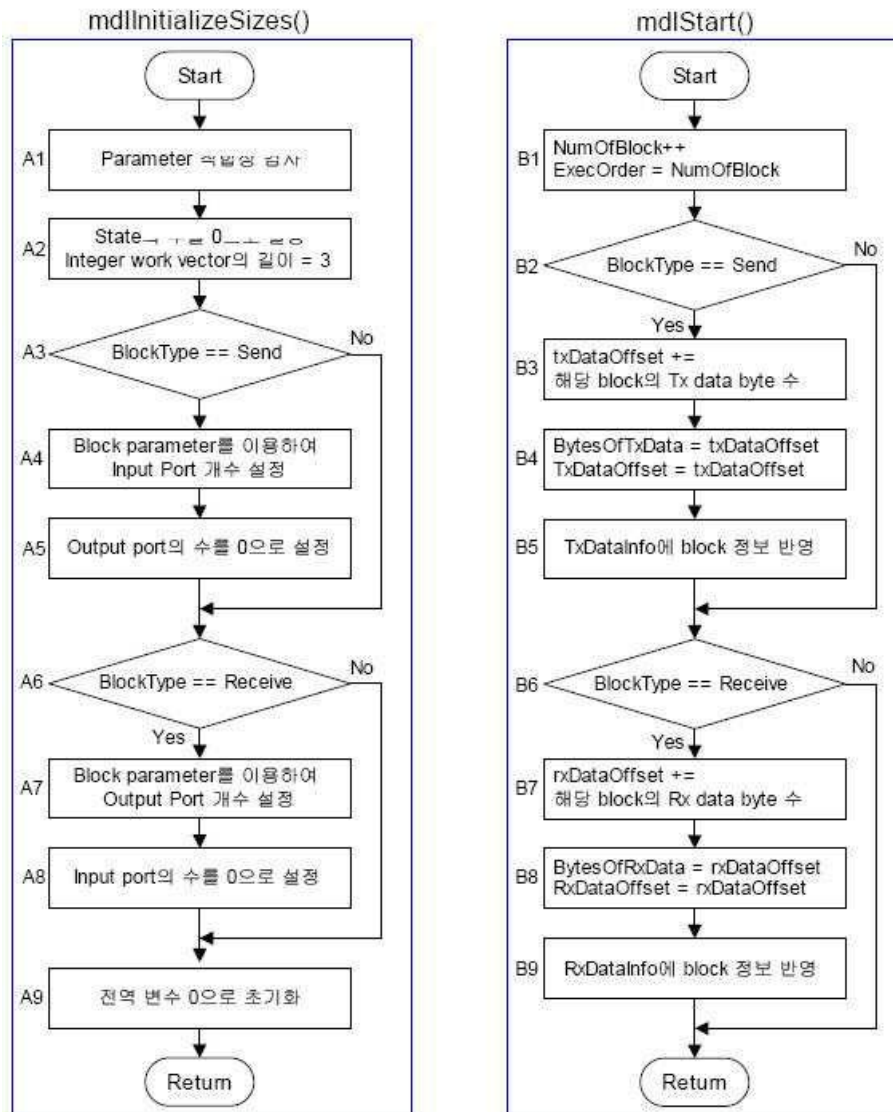
도면23



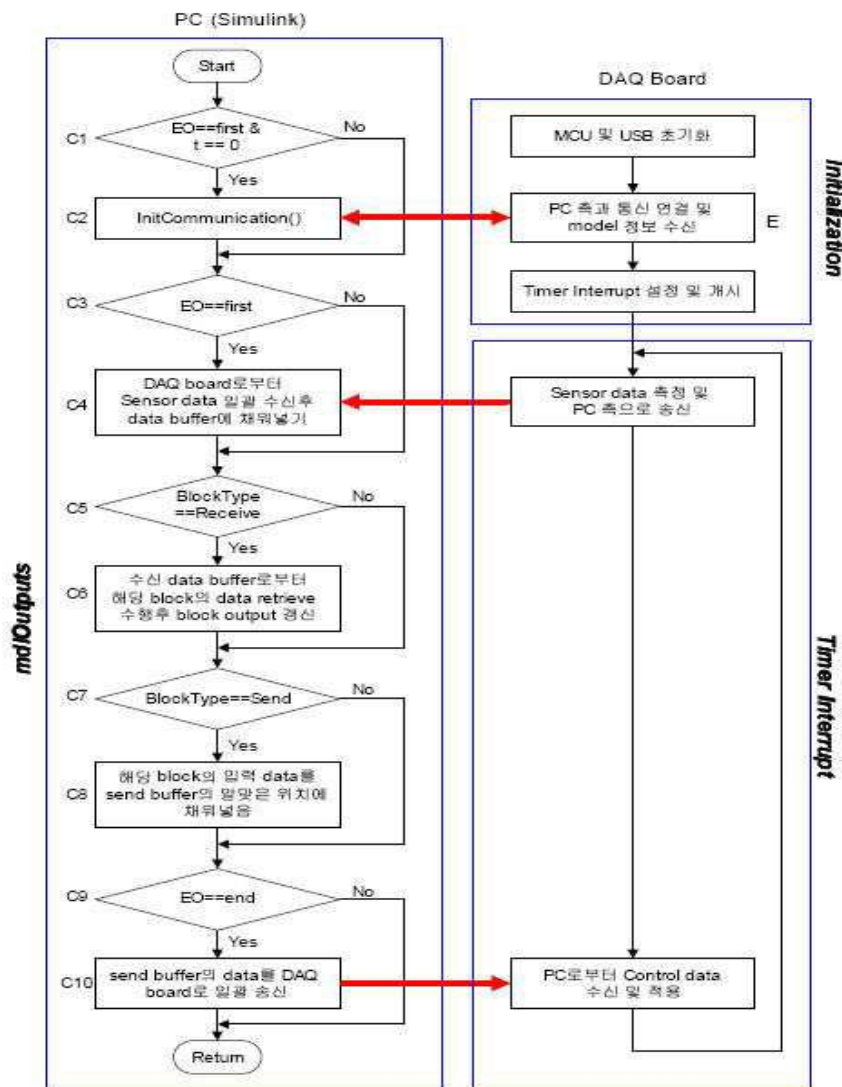
도면24



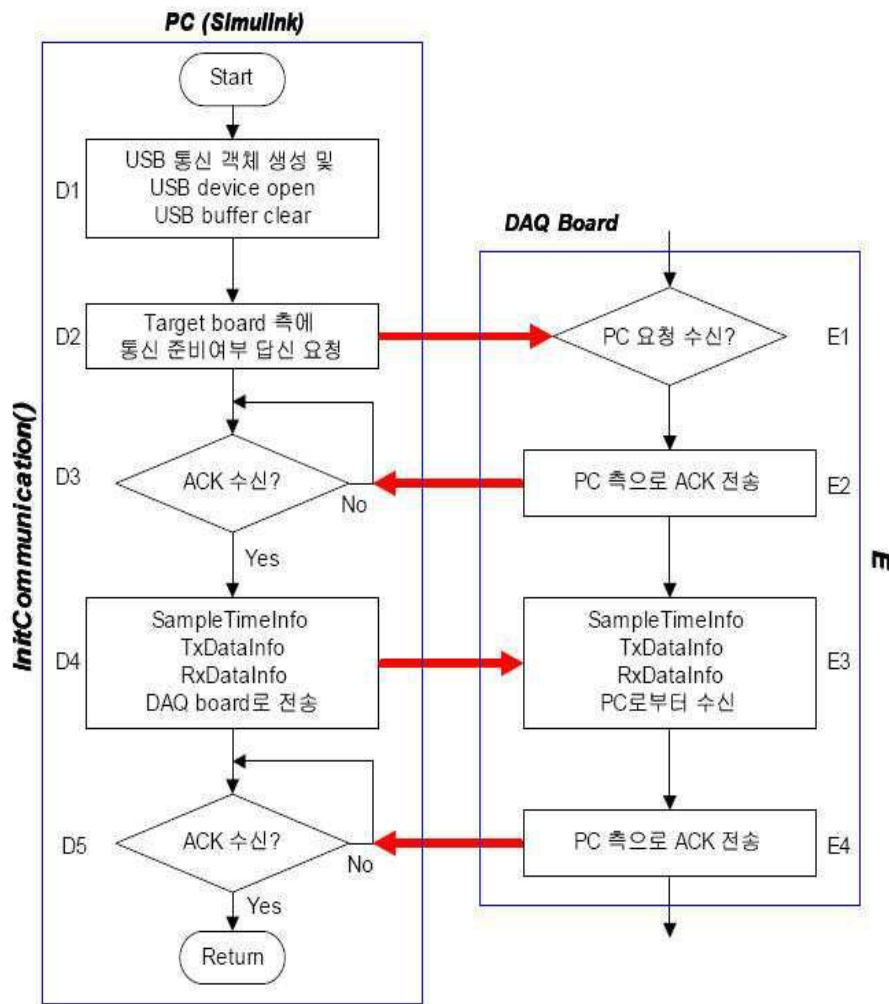
도면25



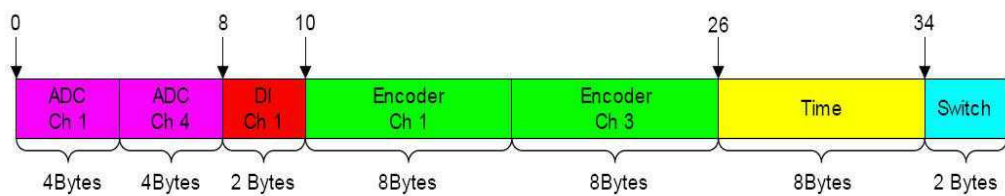
도면26



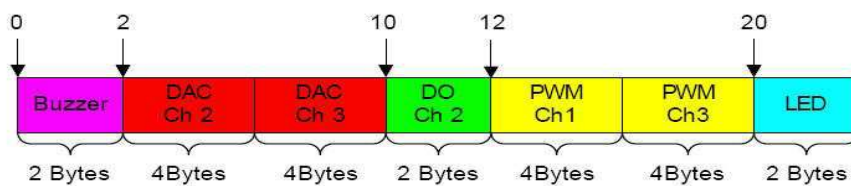
도면27



도면28



도면29



도면30

RCB Block	ExecOrder	RxDataOffset	TxDataOffset
ADC	1	0	-
Buzzer	2	-	0
DAC	3	-	2
DI	4	8	-
DO	5	-	10
ENC	6	10	-
PWM	7	-	12
LED	8	-	20
Time	9	20	-
Switch	10	34	-

도면31

RxDataInfo		TxDataInfo	
ADC CH1	0x20	Buzzer	0xC0
ADC CH4	0x23	DAC CH2	0x91
Digital Input CH1	0x30	DAC CH3	0x92
Encoder CH1	0x10	Digital Output CH2	0xB0
Encoder CH3	0x12	PWM CH1	0x80
Time	0x00	PWM CH3	0x82
Switch	0x40	LED	0xA0