실시간 운영체제와 PLC 어플리케이션의 통합 테스트를 위한 테스트 드라이버 생성 방안

장진아°, 성아영, 최병주
이화여자대학교 컴퓨터학과
{jajang°, aysung}@ewhain.net, bjchoi@ewha.ac.kr

Test Driver Generation for Integration Test between Real-Time Operating System and PLC Application

Jina Jang°, Ahyoung Sung, Byoungju Choi
Dept. of Computer Science and Engineering, Ewha Womans University

요 약

PLC 프로세서 모듈의 시스템 태스크는 RTOS 커널 및 PLC 어플리케이션과 유기적으로 연관되어 있기 때문에 단독으로 테스트 할 수 없고, 어플리케이션을 테스트 드라이버로 활용해야 한다. 본 논문에서는 RTOS와 PLC 어플리케이션의 통합 테스트 단계에서, 시스템 태스크를 테스트 하기 위한 테스트 드라이버를 생성하는 방안을 기술하고, 이를 원자력 발전소의 PLC 프로세서 모듈에 적용한 실험 결과를 기술한다.

1. 서 론

PLC (Programmable Logic Controller)는 원자력 발전소 제어기기, 항공 및 철도 제어 시스템, 공장의 생산 라인 등과 같이 실시간 처리가 요구되는 산업에서 널리 이용되고 있는 임베디드 시스템 (Embedded System)이다 [1]. PLC에 탑재되는 RTOS (Real Time Operating System)는 PLC 어플리케이션을 받아 전체 시스템을 운영한다. 이때, PLC 어플리케이션이란, PLC에 탑재되는 어플리케이션으로서, IEC (International Electro technical Commission) 61131-3에 정의된 FBD (Function Block Diagram), LD (Ladder Diagram), SFC (Sequential Function Chart)와 같은 프로그래밍 언어를 사용하여 작성한다 [2].

원자력 발전소 제어 시스템으로 사용하는 PLC의 경우, PLC에 대한 안전성 확보는 매우 중요하다. RTOS 및 PLC 어플리케이션과 같은 PLC에 탑재되는 임베디드 소프트웨어는 한번 탑재 한 후, 탑재한 소프트웨어에 대한 수정이 용이하지 않기 때문에 신뢰도를 높이는 다양한 테스트가 필수적이다.

RTOS는 커널 (Kernel)과 커널 상에서 동작하는 시스템 태스크 (System Task)로 이루어진다. 시스템 태스크란, 커널 위에서 동작하는 실행 가능한 소프트웨어로써, 운영체제의 기능을 담당한다. 이러한 시스템 태스크는 커널, 하드웨어 장치, PLC 어플리케이션과 밀접하게 유기적으로 연관되어 (Tightly Coupled) 있다.

RTOS는 PLC 어플리케이션을 하나의 태스크로 받아 들인 후, 이를 다른 시스템 태스크들과 함께 운영한다. 즉, 어플리케이션 태스크를 받아 처리하는 RTOS 시스템 태스크의 경우, 커널 및 PLC 어플리케이션 태스크와 유기적으로 연관 되어 있기 때문에, 시스템 태스크를 단독으로 테스트 할 수 없다. 따라서 이러한 시스템 태스크를 테스트 하기 위한 별도의 프로그램이 필요하다.

본 논문에서는 RTOS와 PLC 어플리케이션의 통합 테스트 단계에서, 시스템 태스크를 테스트 하기 위한 테스트 드라이버 (Test Driver)의 생성 방안에 대해 기술하고자 한다. 테스트 드라이버란, 시스템 및 시스템 컴포넌트를 테스트 하는 환경의 일부분으로 테스트를 지원하기 위해 생성된 별도의 프로그램을 의미한다. [3]. 그리고 제안한 방안을 PLC 프로세서 모듈에 적용한 실험 결과를 기술한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 대상으로 하는 원자력 발전소의 PLC 프로세서 모듈 (Processor Module)에 대해 설명하고, 3장에서는 테스트 드라이버 생성에 대해 기술하고, 4장에서는 PLC 프로세서 모듈에 적용한 사례에 대해 기술하고, 5장에서는 결론 및 향후 과제를 기술한다.

2. PLC 프로세서 모듈

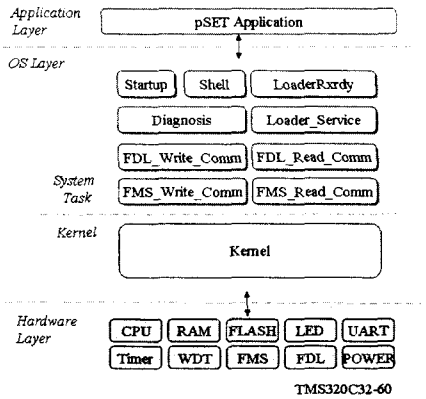
2.1 PLC 프로세서 모듈의 구조

본 장에서는 본 논문에서 대상으로 하는 원자력 발전소의 PLC 프로세서 모듈 [4]을 기술한다.

[그림 1]에서 보는 것과 같이, PLC 프로세서 모듈은 일반적인 임베디드 시스템과 마찬가지로 하드웨어 계층, 운영체제 계층, 어플리케이션 계층으로 구성한다. RTOS의 경우 커널 계층과 시스템 태스크 계층으로 구분한다 [5].

하드웨어 계층 (Hardware Layer)은 Texas Instruments사의

TMS320C32-60 DSP (Digital Signal Processor) [6] 보드를 기반으로 하며, 그 위에 장착된 CPU, RAM, LED, UART, 타이머 (Timer) 등과 같은 물리적인 하드웨어 장치로 구성한다. 운영체제 계층 (OS Layer)은 PLC 프로세서 모듈에 대한 제어 및 운영을 담당하며, uC/OS기반의 RTOS 커널 [7]과 9개의 시스템 태스크로 이루어진다. 어플리케이션 계층 (Application Layer)은 FBD를 기반으로 작성한 원자력 발전소 보호계통 소프트웨어로 이루어진다.



[그림 1] PLC 프로세서 모듈의 구조

커널은 태스크 관리, 태스크간 통신, 스케줄링, 시간 관리 등의 역할만을 수행하고, 9개의 시스템 태스크가 시스템을 운영하는데 필요한 기능을 담당한다. [그림 1]에서 보는 것과 같이, 시스템 태스크에는 시작 태스크 (Startup Task), 쉘 태스크 (Shell Task), 진단 태스크 (Diagnosis Task), 로더 태스크 (LoaderRxrdy Task, Loader_Service Task), 통신 태스크 (FDL_write_comm Task, FDL_read_comm Task, FMS_write_comm Task, FMS_read_comm Task)가 존재한다 [4].

시작 태스크는 시스템 초기화, 메일박스· 메시지큐· 세마포어 생성, 쉘 태스크 생성하고 멀티 태스킹을 통해 시스템을 시작한다. 쉘 태스크는 진단 태스크, 로더 태스크, 통신 태스크를 생성하고, PLC의 LED를 제어 및 디스플레이 (Display) 한다. 진단 태스크는 10ms의 주기로 자가진단을 수행한다. 로더 태스크는 어플리케이션과의 통신을 담당하며, 통신 태스크는 통신 장치와의 통신을 담당한다.

2.2 시스템 태스크와 PLC 어플리케이션의 통합

[그림 1]에서 보는 것과 같이, PLC 프로세서 모듈은 여러 계층이 유기적으로 연결된 임베디드 시스템으로, 다음과 같이 각 계층 간에서 통합이 이루어진다.

- 시스템 태스크와 어플리케이션의 통합
- 시스템 태스크와 시스템 태스크의 통합
- 시스템 태스크와 커널의 통합
- 시스템 태스크와 하드웨어 장치의 통합
- 커널과 하드웨어 장치의 통합

본 논문에서는 시스템 태스크와 어플리케이션의 통합 단계에 초점을 둔다. PLC 프로세서 모듈의 로더 태스크는 PLC 어플리케이션을 태스크로 받아들인다. 이렇게 받아들인 어플리케이션 태스크는 다른 시스템 태스크들과 함께 운영되기 때문에, 어플리케이션과 RTOS 사이에는 반드시 인터페이스(Interface)가 존재한다.

PLC 어플리케이션과 RTOS는 RS-232C (Recommended Standard-232C)를 기반으로 통신한다. RS-232C는 EIA (Electronic Industries Alliance)에서 정의한 통신 프로토콜 (Protocol)로 [8], 외부와 자료를 주고 받기 위하여 국제적으로 표준화한 데이터 통신 규격의 하나이다.

로더 태스크는 RS-232C의 표준 프로토콜을 준수하도록 작성되었기 때문에, RTOS와 어플리케이션 사이의 인터페이스는 다음과 같은 전역 변수를 통하여, 일정한 형식을 가진다 [9].

- *rx_symbol_name*
  '*rx*'가 붙은 전역 변수로서, RS-232C를 통해 로더 태스크가 PLC 어플리케이션을 받아들이는데 사용된다.

- *tx_symbol_name*
  '*tx*'가 붙은 전역 변수로서, RS-232C를 통해 로더 태스크가 PLC 어플리케이션에게 데이터를 전송하는데 사용된다.

3. FBD를 이용한 테스트 드라이버 생성

본 장에서는 로더 태스크를 테스트 하기 위한, 테스트 드라이버의 생성 방안에 대해 기술한다. 이 때, 테스트 드라이버란, 로더 태스크와 같이 단독으로 테스트 할 수 없는 소프트웨어를 테스트 하기 위해서 작성된 별도의 프로그램을 의미한다.

로더 태스크의 *rx_symbol_name*과 *tx_symbol_name*은 RTOS와 PLC 어플리케이션을 함께 실행시켜야 활성화된다. 따라서, 로더 태스크의 *rx_symbol_name*과 *tx_symbol_name*을 테스트 하기 위해서는 PLC 어플리케이션이 필요하다. 즉, 이러한 인터페이스를 테스트 하기 위해 어플리케이션을 테스트 드라이버로 활용하고자 한다.

PLC 프로세서 모듈은 FBD를 기반으로 PLC 어플리케이션을 작성하므로, 본 논문에서는 FBD를 기반으로 테스트 드라이버를 생성한다.

FBD를 이용한 테스트 드라이버의 생성 방안은 [그림 2]에서 보는 것과 같이 FBD 작성, FBD 컴파일 및 링크, FBD에 대한 실행 설정으로 이루어진다.

FBD를 이용한 테스트 드라이버 생성

Step 1. FBD 작성
- **Step 1.1** FBD를 작성하는데 필요한 변수를 추가. 이때, 변수 명, 변수 타입, 변수 크기와 같은 변수 정보 설정.
- **Step 1.2** 함수 블록 추가 및 Step1.1에서 정의한 변수를 활용하여 함수 블록의 입출력 변수로 설정.
- **Step 1.3** 함수 블록들 사이의 연산을 정의.

Step 2. FBD 컴파일 및 링크
- **Step 2.1** FBD 컴파일 및 링크.
- **Step 2.2** FBD 실행파일 생성.

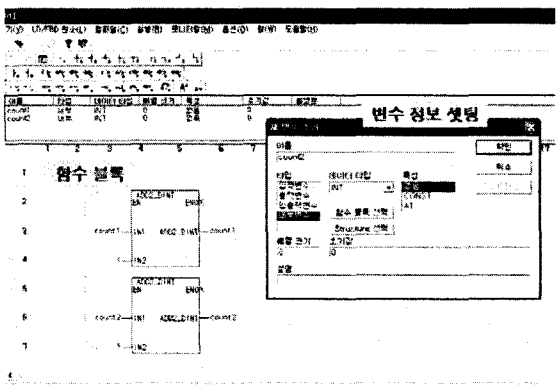
Step 3. FBD 에 대한 실행 설정
- **Step 3.1** RS-232C 통신을 위해 실행파일의 스캔 타임(Scan Time)을 설정
- **Step 3.2** RTOS 가 관리할 실행파일의 상태 설정

[그림 2] FBD를 이용한 테스트 드라이버 생성

■ Step 1. FBD 작성

FBD는 전자회로 다이어그램 (Electrical Circuit Diagram)과 유사한 형태의 도식화된 (Graphical) 언어로써, 단위 기능을 수행하는 각각의 FB들의 네트워크로 표현된다. FBD는 PLC 시스템을 함수 블록 (Function Block) 간의 정보 흐름으로 나타낸다 [2].

[그림 3]은 Step 1.1 ~ Step 1.3에 따라 작성한 FBD의 예를 보여준다. [그림 3]에서 작성한 FBD는 함수 블록 및 변수를 이용하여 덧셈 연산을 수행한다.



[그림 3] 덧셈 연산을 수행하는 FBD의 예

■ Step 2. FBD 컴파일 및 링크

FBD를 컴파일하고 링크하여 FBD에 대한 실행파일을 생성한다.

■ Step 3. FBD 에 대한 실행 설정

Step 2에서 생성한 실행파일과 RTOS를 함께 연동하기 위해서는 실행파일에 대한 실행 정보를 설정해야 한다. 실행파일의 조건으로는 스캔타임 (Scan Time)과 RTOS에서의 수행상태가 있다.

실행파일을 PLC 프로세서 모듈로 다운로드 할 때 사용되는 RS-232C 통신을 위해 실행파일이 실행될 스캔 타임을 설정한다. 또한 RTOS가 실행파일을 관리하기 위해 RTOS에서 수행할 상태를 결정한다.

4. 적용 사례

본 장에서는 원자력 발전소 PLC 프로세서 모듈의 RTOS와 PLC 어플리케이션의 통합 테스트 단계에, FBD를 기반으로 작성한 테스트 드라이버를 적용한 사례에 대해 기술한다.

4.1 테스트 사례

로더 태스크가 PLC 어플리케이션과 가지는 인터페이스를 활성화 하기 위해서, FBD를 이용하여 어플리케이션을 [표 1]과 같이 작성하였다.

본 논문에서는 어플리케이션을 다양화 하기 위해 어플리케이션의 크기를 조작하였다. PLC 어플리케이션에서 구현한 내용 자체보다는, RTOS가 어플리케이션을 관리하는데 사용하는 어플리케이션 속성과 관계된 정보에 초점을 두었기 때문에, '덧셈 연산을 수행하는' 함수 블록만을 추가하여 크기를 조작함으로써, 어플리케이션을 다양화 하였다.

[표 1] 생성한 어플리케이션 목록

파일 명	라인 수	파일 크기	구현 언어
App.1	4	1 KB	FBD
App.2	301	40 KB	FBD
App.3	1501	197 KB	FBD
App.4	3001	394 KB	FBD
App.5	4501	592 KB	FBD
App.6	6001	700 KB	FBD
App.7	7501	971 KB	FBD

[표 1]은 크기를 다양화해서 생성한 7개의 PLC 어플리케이션에 대한 전체 라인 수, 파일 크기, 구현 언어를 나타낸다. PLC 프로세서 모듈에서 어플리케이션을 다운로드 받는 RAM 공간 제한으로, 크기가 약 1MB를 초과 할 수 없기 때문에, 최소 1KB부터 최대 971KB까지의 총 7개의 어플리케이션을 작성하였다.

7개의 어플리케이션에 대한 실행 정보는 다음과 같다. 스캔 타임은 10ms, 어플리케이션의 상태는 다운로드로 설정하였다.

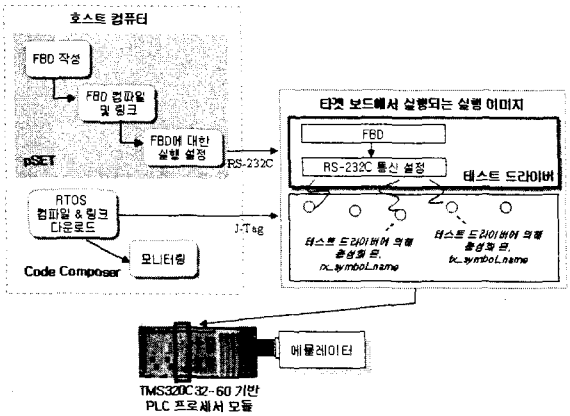
4.2 테스트 환경

본 논문에서 RTOS와 PLC 어플리케이션의 테스트를 수행한 환경은 [그림 3]과 같다. [그림 3]에서 보는 것과 같이,

호스트 컴퓨터에는 PLC 어플리케이션 개발 도구인 pSET (POSAFE-Q Software Engineering Tool)과 통합 개발 환경인 Code Composer [10]가 존재한다. 하드웨어는 호스트 컴퓨터와 PLC 프로세서 모듈을 구성하는 TI TMS320C32 기반의 타겟 보드, TI사의 보드를 지원하는 에뮬레이터, RS-232C와 J-Tag 케이블이 있다.

pSET은 FBD를 기반으로 어플리케이션을 작성하고 컴파일과 링크를 수행하여 실행 파일을 생성한다. 생성한 어플리케이션 프로그램의 실행파일에 RS-232C 시리얼 통신을 위한 스캔 타임을 설정하고, RTOS가 관리할 어플리케이션 태스크의 상태를 설정한다. 이러한 과정을 통해 pSET에서 작성한 FBD는 RS-232C 통신 설정을 거쳐 테스트 드라이버로써 활용된다.

Code Composer는 TI사의 전용 타겟 보드에 탑재될 RTOS 프로그램을 작성하고, 모니터링과 디버깅을 지원하는 통합 개발 환경이다. Code Composer는 RTOS 프로그램에 대한 컴파일 및 링크를 통해 실행파일을 생성하고, 생성된 실행파일을 테스트 드라이버와 함께 타겟 보드인 PLC 프로세서 모듈로 탑재 한다.



[그림 3] 테스트 환경

4.3 테스트 결과

본 논문에서는 타겟 보드가 맞물린 실행 환경에서 실시간 모니터링 (Run-Time Monitoring)을 이용하여 테스트를 수행하였다 [11, 12]. 이때, 테스트는 실행 중인 RTOS를 브레이크 포인트 (Break Point)를 이용하여 정지 시킨 후, 해당 위치의 현재 값을 모니터링 함으로써 수행하였다. 어플리케이션에 의해 활성화된 로더 태스크의 *rx_symbol_name*과 *tx_symbol_name*에 브레이크 포인트를 설정하고, PLC 프로세서 모듈이 실행되는 동안 이 브레이크 포인트에서의 결과 값을 모니터링 하였다. 모니터링 한 결과가 예상 출력과 같으면 통과 (Pass)로 판단하고, 그렇지 않으면 실패 (Fail)로 판단하였다. 이때, 예상 출력은 RS-232C 프로토콜에 따라 실행 되는, RTOS 상의 실행 경로 및 각 실행 경로에서 *rx_symbol_name*과 *tx_symbol_name*이 가져야 하는 값을 근거로 하였다.

본 실험에서는, RS-232C를 기반으로 PLC 어플리케이션이 시리얼 통신을 완료한 직후, 생성한 7개의 어플리케이션을 활용하여 로더 태스크가 가지는 인터페이스를 모두 테스트 하였다. 이때, 로더 태스크가 PLC 어플리케이션과 가지는 인터페이스는 [표 2]에서 보는 것과 같이 *rx_mode, rx_header[], rx_checksum, rx_bcs, rx_text[], rx_test_ptr, tx_return* 이다.

[표 2] FBD 크기에 따른 테스트 결과

인터페이스	App. 1	App. 2	App. 3	App. 4	App. 5	App. 6	App. 7
rx_mode	EOT	EOT	EOT	EOT	EOT	EOT	EOT
rx_header[0]	E	E	E	E	E	E	E
rx_header[1]	D	D	D	D	D	D	D
rx_checksum	1720	1535	1685	1744	1549	1864	1669
rx_bcs	1720	1535	1685	1744	1549	1864	1669
rx_text_ptr	60	60	60	60	60	60	60
rx_text[0]	0	0	0	0	0	0	0
rx_text[1]	0	0	0	1	2	2	3
rx_text[2]	0	38	188	119	51	238	170
rx_text[3]	244	20	20	148	20	148	20
rx_text[4]	0	0	0	0	0	0	0
tx_return	ACK	ACK	ACK	ACK	ACK	ACK	ACK

[표 2]에서는 RTOS와 PLC 어플리케이션의 통합 테스트 단계에, 어플리케이션이 PLC 프로세서 모듈에 다운로드를 완료한 직후, 각 인터페이스에서의 테스트 결과를 기술한다. [표 2]의 각 인터페이스와 인터페이스가 가지는 테스트 결과값의 의미는 아래와 같다.

rx_mode, *rx_header[]*, *tx_return*은 RS-232C 프로토콜에 따라, 작성한 어플리케이션의 크기에 상관 없이 일정한 값을 가진다. *rx_mode* 는 어플리케이션과 RTOS 사이의 RS-232C 통신을 완료했다는 의미의 'EOT' (End Of Text)를 의미한다. *rx_header[]*이 가지는 ' E' 와 ' D' 는 PLC 어플리케이션이 현재 다운로드 상태였다는 것을 의미한다. *tx_return*은 통신이 무사히 완료되었다는 ACK을 의미한다.

*rx_checksum*과 *rx_bcs*는 송수신한 데이터의 무결성을 보장하기 위해 사용하는 변수이며, *rx_text_ptr, rx_text[]*는 PLC 프로세서 모듈에 전송 받은 데이터를 저장하기 위해 사용하는 변수이다. *rx_checksum, rx_bcs, rx_text_ptr, rx_text[]*는 어플리케이션의 크기에 따라 각각 다른 값을 가진다.

5. 결론 및 향후 과제

RTOS 는 PLC 와 같은 임베디드 시스템에 탑재되어, 어플리케이션을 동작시킴으로써 전체 시스템을 운영한다. 원자력 발전소 제어와 같은 PLC 에 탑재되는 RTOS 는 하드웨어 및 어플리케이션과 유기적으로 연관되어 있기 때문에, 높은 신뢰도를 요구하는 소프트웨어임에도 불구하고, 테스트가 용이하지 않다.

특히, PLC 어플리케이션을 RS-232C로부터 받아, 이를 관리하는 RTOS 시스템 태스크의 경우, 커널 및 PLC 어플리케이션 태스크와 유기적으로 연관 되어 있기 때문에, 그 시스템 태스크를 단독으로 테스트 할 수 없다.

본 논문에서는 시스템 태스크를 테스트 하기 위한 테스트 드라이버를 생성하는 방안을 기술하고, 제안한 방안에 따라 테스트 드라이버를 생성하여 이를 원자력 발전소 PLC에 탑재되는 RTOS와 PLC 어플리케이션의 통합 테스트 단계에 적용하였다.

향후, 본 논문에서 제안한 방안을 PLC에 존재하는 다른 계층간의 통합 테스트에 확장 적용할 예정이다.

6. 참고 문헌

[1] A. Mader, "A Classification of PLC Models and Applications", in the Proc. of International Workshop on Discrete Event Systems -- *Discrete Event Systems, Analysis and Control*, Kluwer Academic Publishers, pp.239-247, 2000.

[2] IEC, *International Standard for Programmable Controllers: Programming Languages, Technical Report IEC 1131 part 3*, IEC (International Electro technical Commission), 1993.

[3] D.J. Panzl, "Automatic Software Test Driver", *IEEE Computer*, Vol.11, pp.44~50, 1978.

[4] KNICS-PLC-SDS331-01, *Software Design Specification for the PLC Processor Module*, KAERI (Korea Atomic Energy Research Institute), 2006.

[5] A. Jerraya and W. Wolf, "Hardware/Software Interface Codesign for Embedded Systems", *IEEE Computer*, Vol.38, pp.63~69, 2005.

[6] TMS320C32 Digital Signal Processor available in http://www.ti.com/, Texas Instrument, 1998.

[7] J.J Labrosse, *MicroC/OS-II, The Real-Time Kernel*, CMP Books, 1999.

[8] EIA/TIA-232-C, *Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange*, EIA (Electronic Industries Alliance), 1991.

[9] A. Sung, J. Jang and B. Choi, "Fault-Based Interface Testing Between Real-Time Operating System and Application", Mutation2006, USA, submitted.

[10] SPRU296, Code Composer User's Guide, Texas Instrument, 1999.

[11] S. E. Chodrow, F. Jahnian, and M. Donner, "Run-Time Monitoring of Real-Time Systems", in the Proc. of Run-Time Systems Symposium, IEEE, pp.74-83, 1991.

[12] S. Ricardo and Jr. J. R. de Almeida, "Run-Time Monitoring for Dependable Systems: an Approach and a Case Study", in the Proc. of International Symposium on Reliable Distributed System, IEEE, pp.41-49, 2004