임베디드 시스템 소프트웨어 개발을 위한 정형적 접근

이수영, 김진현, 최진영
고려대학교 컴퓨터학과
e-mail : {sylee, jhkim, choi}@formal.korea.ac.kr

Formal Approach for Embedded System Software Development

Su-Young Lee, Jin-Hyun Kim, Jin-Young Choi
Dept. of Computer Science, Korea University

요 약

임베디드 시스템 소프트웨어 개발과정에서, 자연어로 작성된 요구명세와 소프트웨어 엔지니어와 하드웨어 엔지니어 사이의 서로 다른 언어와 개발도구의 차이로 인해 많은 문제들이 있어왔다. 즉, 개발자의 실수로 설계가 잘못 명세 되었거나 요구명세와 실제 구현된 시스템 소프트웨어의 인터페이스 코드나 요구된 수행이 일치하지 않았다. 이를 해결하기 위해 본 논문에서는 정형기법을 이용하여 요구사항을 명세하고 설계를 검증함으로써 개발자의 실수로 인한 오류를 줄이고 개발된 시스템 소프트웨어의 인터페이스 코드 및 수행이 요구명세를 만족함을 보이도록 정형기법을 이용한 개발 프레임워크를 제안하고자 한다.

1. 서론

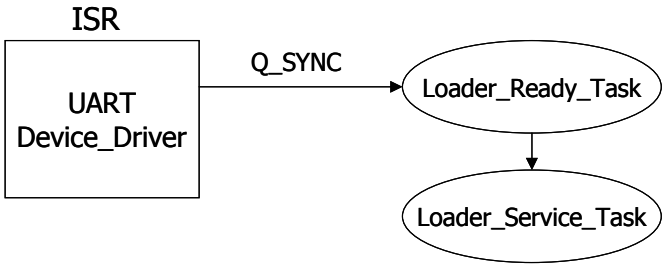
임베디드 시스템은 마이크로프로세서의 90% 이상이 개인 PC 와 같은 범용 컴퓨터가 아닌, 임베디드 시스템에 사용되고 있을 정도로 많이 사용되고 있다. 특히 원자력 발전소 제어시스템이나 자동차의 브레이크 제어 시스템, 항공기 제어 시스템과 같이 고안전성 시스템은 설계자의 조그마한 실수로 막대한 인적, 물적 손실을 유발할 수 있는 분야이다. 따라서 임베디드 시스템의 설계 단계에서부터 임베디드 시스템 소프트웨어의 신뢰성을 보장할 수 있도록 엄격한 개발 과정이 요구된다.

본 논문은 시스템 레벨 관점에서 임베디드 시스템 소프트웨어를 구현한 모델을 검증하도록, 정형기법을 적용한 시스템 소프트웨어 개발 방법론을 제시한다. 즉, 시스템 소프트웨어의 설계 명세를 정형적으로 명세하고, 설계한 명세에 대해서 주어진 요구 조건을 만족하는지 시뮬레이션과 검증기법을 이용해서 확인 및 검증한다. 이렇게 확인, 검증된 설계 명세를 바탕으로 실제 시스템을 구현함으로서 시스템의 오류를 줄일 수 있고, 임베디드 시스템 소프트웨어의 신뢰성을 보장해 줄 수 있다.

본 논문은 정형기법 도구인 Esterel[1]을 이용하여 Esterel 언어로 시스템 소프트웨어의 설계를 명세하고, 시뮬레이션 도구인 XES[2]와 정형검증 도구인 XEVE[3]를 이용하였다. 또한, Esterel 의 컴파일러인 Esterel toolset 은 Esterel 언어로 작성된 설계 명세로부터 C 나 VHDL 코드를 자동생성 해준다. 본 논문은 다음과 같이 구성되어 있다. 2 장은 임베디드 시스템 소프트웨어의 요구사항에 따른 설계 명세를 Esterel 을 이용하여 작성하는 절차에 대해 설명하고, 3 장에서는 Esterel 의 검증도구인 XES 와 모델체커인 XEVE 를 이용하여 임베디드 시스템 소프트웨어가 검증 특성들을 만족하는지를 확인해본다. 4 장에서는 시뮬레이션과 검증을 통해 요구 조건의 만족 여부가 확인된 임베디드 시스템 소프트웨어를 실제로 구현하고 타겟보드인 80c196 에 포팅하여 실제 동작함을 확인한다. 마지막으로 5 장에서는 결론을 맺고 향후 연구 계획에 대해 소개한다.

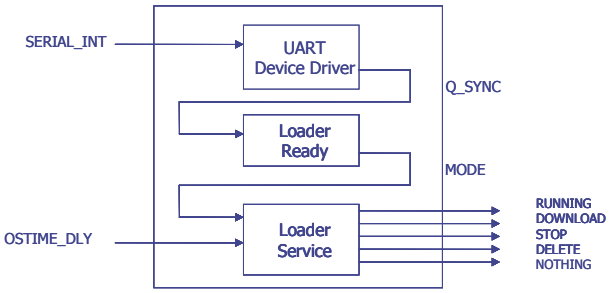
2. 임베디드 시스템 소프트웨어의 설계 명세

본 논문에서는 시스템 레벨 관점에서 시스템 소프트웨어를 검증하기 위해 상위 레벨에서 여러 개의 태스크들을 스케줄링하도록 설계하였다. 임베디드 시스템 개발 방법론을 적용한 UART 디바이스 드라이버는 비동기 시리얼 통신의 하위레벨 작업을 핸들링하는 통합된 서킷인 UART 의 드라이버이다. 비동기 통신은 UART 를 통해 수행하는데 데이터를 송수신할 때 프로그램은 단순히 UART 로부터 한 바이트의 데이터를 읽거나 쓰는 동작을 한다. 본 논문에서는 시스템 레벨의 검증을 위해 UART 디바이스 드라이버는 호스트로부터 데이터를 수신하고 버퍼가 찼을 때, 인터럽트를 발생하여 태스크를 스케줄링하도록 모델링한다. UART 디바이스 드라이버의 동작 모델은 실시간 운영체제인 uC/OS-II[4]에서 실행 가능한 태스크의 형태로 구현하기 위해 UART 디바이스 드라이버 모듈과 실제 운영체제에서 실행되는 Loader_Ready 태스크 모듈, Loader_Service 태스크 모듈을 Esterel 을 이용하여 명세하였다. (그림 1)과 같이 UART 디바이스 드라이버는 인터럽트 발생시 Loader_Ready 태스크를 스케줄링하고 Loader_Ready 태스크는 UART 버퍼로부터 수신한 데이터를 복사 후 Loader_Service 태스크를 스케줄링한다.



(그림 1) 시스템 레벨에서의 UART 디바이스 드라이버 설계 모델

UART 디바이스 드라이버는 하드웨어의 인터럽트 신호를 입력 언어인 Esterel 을 이용하여 입력 신호로 받아들이도록 인터럽트 처리 루틴에서 Loader_Ready 태스크를 스케줄링하는 일을 수행하도록 하였다.



(그림 2) 시스템 레벨에서의 UART 디바이스 드라이버 동작 모델

(그림 2)의 UART 디바이스 드라이버의 입력, 출력 신호는 다음과 같다.

Input Signals

- SERIAL_IN : 호스트와 UART 버퍼와의 데이터 송수신 완료 시 Interrupt 발생함을 SERIAL_INT signal 로 입력 받는다.
- OSTIME_DLY : 1 tick 마다 Loader service task 가 수행하도록 time delay 시그널로 입력 받는다.

In Relation to Output Signals

- RUNNING : mode 가 1 일 때 RUNNING 임을 알린다.
- STOP : mode 가 2 일 때 STOP 임을 알린다.
- DOWNLOAD : mode 가 3 일 때 DOWNLOAD 임을 알린다.
- DELETE : mode 가 4 일 때 DELETE 임을 알린다.
- NOTHING : mode 가 0 일 때 즉, MODE 가 어떤 명령어로도 설정되지 않았을 때 NOTHING 임을 알린다.

InputOutput Signals

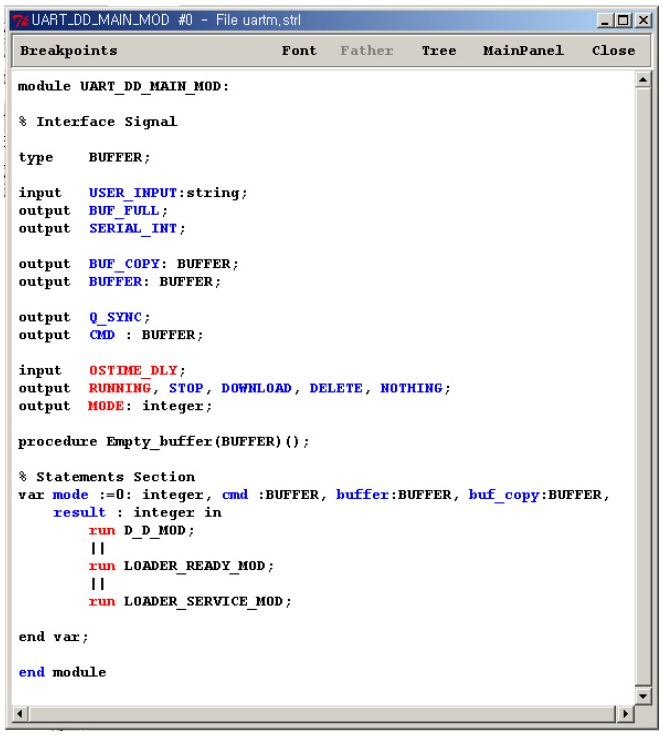
- Q_SYNC : SERIAL_INT 발생시 UART 디바이스 드라이버 모듈이 Q_POST 상태의 Q_SYNC 를 발생하면 Loader_Ready 태스크는 Q_PEND 상태로 Q_SYNC 를 기다렸다가 시그널이 발생하면 스케줄링된다.

Output Signals

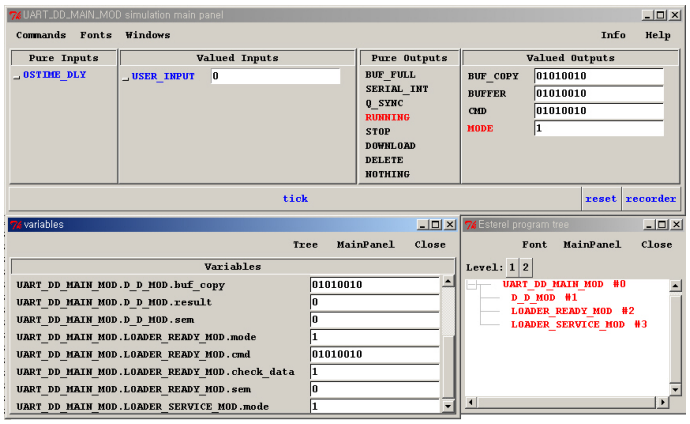
- MODE : Queue 에서 읽어온 buffer 값에 따라 Loader_Service 태스크가 제공할 서비스 모드를 설정한다.

(그림 3)과 같이 UART 디바이스 드라이버는 UART 디바이스 드라이버 모듈, Loader_Ready 태스크 모듈, Loader_Service 모듈로 나뉘어 병행적으로 동작하도록 Esterel 언어로 설계하고 모듈들간에 브로드캐스팅(Broadcasting)되는 시그널들을 통해서 동기화되며 주어진 작업을 수행한다. (그림 4)는 UART 디바이스 드라이버의 설계명세를 Esterel 의 검증 시뮬레이션 도구인 XES 를 이용하여 시스템의 타당성(Validation) 검증을 시행하였다. XES 는 UART 디바이스 드라이버를 유한 상태 기계로 표현한 것으로 인터럽트 발생시 UART 로부터 버퍼값을 읽어들여 수신모드를 설정하고 그에 따른 서비스를 올바르게 수행하는지 시뮬레이션 해보았다. 임의로 입력 신호인 USER_INPUT 으로 UART 의 버퍼에 데이터를 전송하여 buffer full 일 때, 인터럽트를 발생시키는 출력신호인 SERIAL_INT 를 발생하여 Loader_Ready 태스크를 스케줄링한다. Loader_Ready 태스크는 수신모드를 설정하고, Loader_Service 태스크는 수신모드에 따라 서비스를 수행하고 있음을 확인할 수 있었다. 이렇게 명세한

UART 디바이스 드라이버의 설계명세를 가지고 3 장에서 정형 검증해보고자 한다.



(그림 3) UART 디바이스 드라이버 동작 모델의 Esterel 설계 명세



(그림 4) UART 디바이스 드라이버 동작 모델의 검증 (XES)

3. 임베디드 시스템 소프트웨어의 검증

본 논문에서는 조건의 만족 여부를 XEVE 에서의 Input 유무 조건에 따라 출력 신호의 상태를 체크하여 만족 여부를 확인해보고자 한다. Interrupt 발생에 의해 동작하는 태스크 1 이 있고 태스크 1 에 의해 동작하는 태스크 2 가 있고 또 다른 여러 개의 태스크들이 OS 상에서 돌고 있는 시스템을 설계한다고 하자. 이렇게 동작하도록 설계한 설계 명세서는 설계자의 실수로 인해 잘못 명세 되었을 경우, Interrupt 에 의해 동기화가 되어 동작해야만 하는 태스크 1 이 태스크 2 나 또다른 태스크들에 의해 동기화가 되어 동작되는 경우가 발생한다. 설계자의 실수로 인해 잘못 명세된 설계에서 요구하는 사항들이 올바르게 구현되었는지를 검증하기 위해 UART Device Driver 가 만족해야 할 검증 조건들을 주어 XEVE 를 통해 검증하고자 한다.

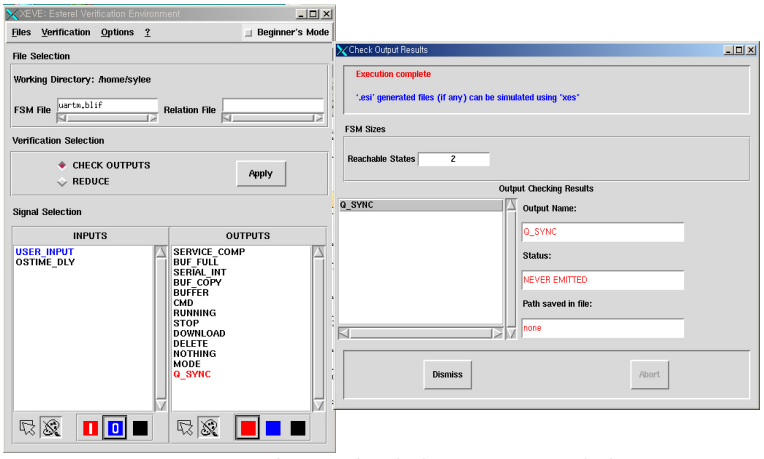
UART 디바이스 드라이버가 갖추어야 할 검증 특성으로 하드웨어에서 발생하는 인터럽트가 소프트웨어에서 제대로 인터럽트를 처리하는지와 인터럽트 루틴에서 수행되는 서비스들이 올바른 행위를 보이는지를 다음과 같이 두 가지 조건을 두어 검증해보았다.

□ Interrupt ∧ □ Q_SYNC (1)

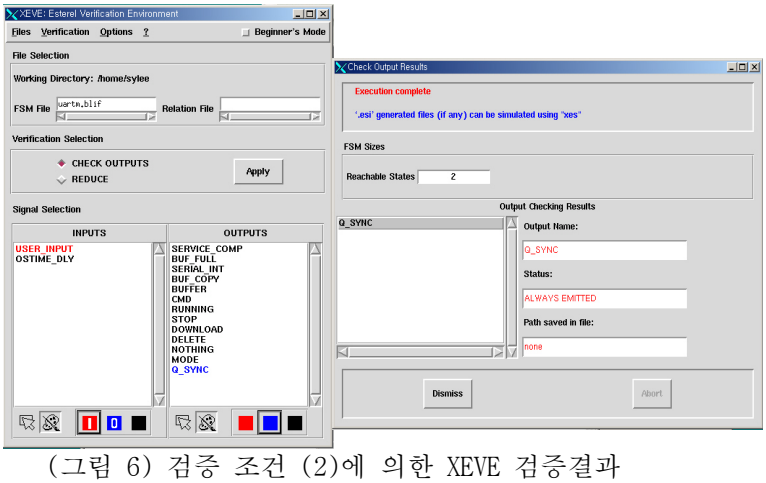
Interrupt -> Q_SYNC (2)

- 검증조건 (1) : 인터럽트 신호의 발생에 의해서만 Loader_Ready 태스크는 동기화가 되어야 하므로, 인터럽트 신호가 발생하지 않으면 Loader_Ready 태스크는 절대로 동기화 되어서는 안된다.(Never emitted)

- 검증조건 (2) : 인터럽트 신호의 발생에 의해서만 Loader_Ready 태스크는 항상 동기화가 되어야만 한다.(Always emitted)



(그림 5) 검증 조건 (1)에 의한 XEVE 검증결과



(그림 6) 검증 조건 (2)에 의한 XEVE 검증결과

우선 설계한 시스템이 만족해야 할 요구 조건을 XEVE에서 INPUT 신호들의 유무로 설정하여 그에 따른 OUTPUT 상태를 관찰한다. 만족하는 경우에는 시스템이 본래 의도대로 제대로 설계되었다는 것을 의미하고, 위반하는 경우는 XEVE 분석에 의한 해당 검증 조건의 반례(counter example)를, 그 발생 가능성에 따라 possibly emitted 또는 never emitted 또는 always emitted 시그널을 출력시키게 된다. 분석된 반례는 다시 시뮬레이터를 통해 그 발생 경로를 추적 후 수정하게 되고, 시스템 모델의 오류 수정 후 시스템의 재검사 작업이 이루어진다. UART 디바이스 드라이버는 검증특성이 요구하는 결과를 만족하고 있다. 그 결과로 검증특성 (1)은 never emitted 시그널을 출력, 검증특성 (2)는 always emitted 시그널을 출력하고 있다. 따라서 시스템이 알고리즘의 원칙대로 동작한다는 사실을 알 수 있고 이를 토대로 시스템 소프트웨어가 올바르게 설계되었음을 확인해 볼 수 있었다.

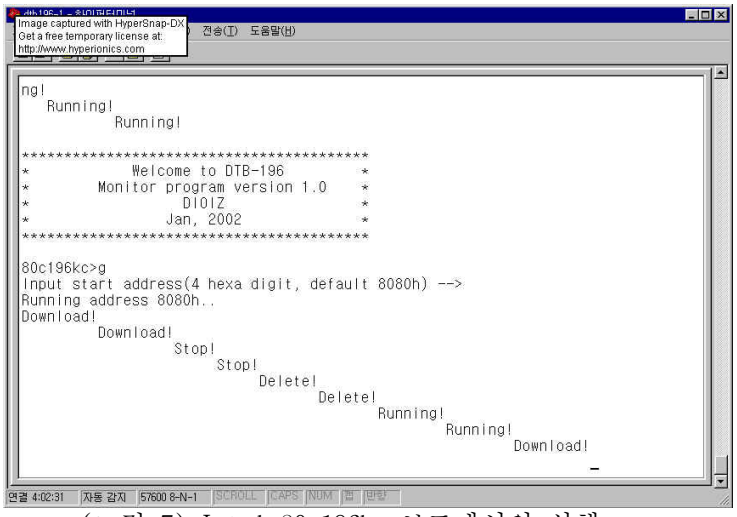
4. 구현 및 실행

본 절에서는 Esterel 설계명세로부터 실행가능한 코드를 자동 생성하여 타겟 보드인 Intel 80c196kc 마이크로프로세서가 탑재된 보드에 포팅 및 실행해보았다. 명세한 Esterel 설계서는 실행보다는 시스템 소프트웨어의 검증을 위해 작성된 설계서이다. 따라서 실제로 타겟 보드에서 실행할 수 있는 코드를 생성하기 위해서는 불필요한 input 값들과 output 값들을 생략하고 실행 구현 가능한 설계서로 작성하였다.

앞절에서 Esterel 을 이용하여 설계한 UART 디바이스 드라이버는 Esterel 의 자동 코드 생성기를 이용하여 자동변환된 C 코드를 얻었다. 시스템 구현을 위한 코드는 (1) UART 디바이스 드라이버 명세로부터 자동생성된 임베디드 코드와 (2) Data-handling 코드로, (1)코드는 태스크 관리 및 자원 사용에 관한 작업을 담당한다. 자동 변환된 C 코드에서 Esterel 의 입력, 출력 시그널은 함수 형태로 생성되므로, 입력, 출력 시그널이 발생하면 C 코드에서는 함수 호출이 되어 함수 안에 구현된 코드를 수행하게 된다. (2)코드는 이 함수 안의 수행되는 코드를 구현한 것이다. 이와 같이 작성한 C 코드를 타겟보드에서 실행할 수 있도록 포팅을 하였고, (그림 7)은 타겟보드인 intel 80c196kc 보드에서 실행한 모습이다.

5. 결론 및 향후 계획

임베디드 시스템 소프트웨어 개발과정에서, 자연어로 작성된 요구명세와 소프트웨어 엔지니어와 하드웨어 엔지니어 사이의 서로 다른 언어와 개발도구의 차이로 인해 많은 문제들이 발견되었다. 즉, 개발자의 실수로 설계가 잘못 명세 되었거나 요구명세와 실제 구현된 시스템 소프트웨어의 인터페이스 코드나 요구된 수행이 일치하지 않았다. 이를 해결하기 위해 본 논문에서는 정형기법을 이용하여 요구사항을 명세하고 설계를 검증함으로써 개발자의 실수로 인한 오류



(그림 7) Intel 80c196kc 보드에서의 실행

를 줄이고 개발된 시스템 소프트웨어의 인터페이스 코드 및 수행이 요구명세를 만족함을 보이도록 정형기법을 이용한 개발 프레임워크를 제안하였다. 이처럼 일반적인 임베디드 시스템 개발 방법론으로는 하드웨어와 소프트웨어를 각각 개발시 발견하기 어려운 에러나 인터페이스상의 문제점이나 시뮬레이션의 문제점들을 정형기법을 적용하여 개발함으로써 하드웨어와 소프트웨어의 co-design 가능성을 보였고, 시스템 소프트웨어의 정확한 검증과 시스템 동작의 정확성을 효율적으로 검증함으로써 정형기법의 적용 가능성을 제시하였다. 그러나 개발 대상을 이해하고 모델링을 거쳐 명세 및 검증에 이르기까지의 과정에서 개발자는 임베디드 시스템에 대한 완벽한 이해를 필요로 한다. 그러므로 요구 명세와 디바이스 드라이버와 같은 하드웨어 의존적인 소프트웨어의 경우, 하드웨어와 임베디드 시스템에 대한 배경지식과 하드웨어와 소프트웨어의 영역을 효율적으로 분할 및 설계한 완벽한 명세가 필요하다고 본다. 향후 연구과제로는 Esterel 을 이용하여 하드웨어를 설계하고 이를 기반으로 실제 하드웨어를 개발할 수 있는 VHDL 코드를 생성하여 실제 임베디드 시스템의 HW/SW 통합과정을 수행하는데 있다.

참고문헌

[1] Gerard Berry. ESTEREL v5 Language Primer Version 5.21 release 2.0. April 6. 2000
[2] G.Berry and The Esterel Team. The Esterel v5_21 System Manual. INRIA France. March. 11. 1999
[3] Amar Bouali. XEVE: an Esterel Verification Environment(Version v1_3) INRIA. France. 1997
[4] J J. Labrosse, "Micro C/OS-II The Real-Time Kernel Second Edition", CMP Books, 2000
[5] F Boussinot. and R. De Simone. "The Esterel Language," *Proc. IEEE,* Vol. 79, No. 9, pp. 1293-1304, Sep. 1991
[6] G K Palshikar, "An Introduction to Esterel", Embedded Systems Programming in CMP Media, 2001
[7] J J. Labrosse, “Embedded Systems Building Blocks Second Edition. Complete and Ready-to-Use Modules in C”, CMP Books, 2002