

전력기기의 임베디드 OS 모듈 구성에 관한 연구

권호철*, 오재훈*, 오성민*, 홍정기*
(주)효성 중공업연구소*

A Study on structure of embedded OS module for intelligent electronic device

Hyo-Chul Kwon*, Jea-Hun Oh*, Sung-Min Oh*, Jung-Ki Hong*
HYOSUNG corporation Power & Industrial Systems R&D Center*

Abstract - Embedded OS is used widely in many case of today's systems. Using of Embedded OS for IED(Intelligent electronic device) is vary useful method of the processing of complex operations. In this paper, we designed embedded OS module for use as a more effective way in IED systems.

2. 본 론

1. 서 론

요즘 Embedded system에서 OS의 사용이 늘어가고 있다. 이것은 비약적인 하드웨어의 발달에 기인함과 동시에 더 많은 기능을 요구하는 사용자들이 늘어가기 때문이기도 하다. 또한, 고객에게 더 빠른 시간 안에 제품을 제공해야 하는 기업 상황이 OS의 사용을 더 부추기고 있다.

Embedded OS의 사용이 늘어남에 따라 더 사용이 편리하고 기능이 확실한 OS의 개발이 진행되고 있으며, 세계적으로 많은 수의 OS가 Embedded system에서 쓰이고 있다.

OS는 Real-Time의 동작을 수행하는 OS로 구분이 가능하다. OS가 Real-Time을 보장 한다는 의미는 정해진 시간 내에 실행되는 Process를 완료할 수 있으며, 또한, Process의 우선순위에 따라 OS에서 스케줄링을 해줄 수 있음을 의미한다.

또한, OS를 구조적인 면에서 Multi Process와 Multi Thread의 두 가지로 분류 할 수 있으며 종류는 다음 표1과 같다.

<표 1> OS의 분류 및 종류

분류	종류	개발사
Multi Process RTOS	QNX	QNX Software system
	OS-9	Microware
	LynxOS	LinuxWorks
	RTLinux	Finite State Machine Labs
	Windows CE	Microsoft
Multi Thread RTOS	OSE	Enea OSE systems
	VRTX	Mentor Graphics
	pSOS	Integrated systems
	uC/OS	Micrium
	Nucleus Plus	Accelerated Technology

이러한 OS의 종류만큼 다양한 Application이 존재할 수 있다. 이런 경우 Application의 모듈화를 통하여 다양한 system에서 통일된 Application이 적용될 수 있도록 software의 설계가 필요하다.

IED(intelligent electronic device)의 경우 외국 유수의 기업들은 OS의 활용에 적극적인 반면 국내 업체들은 아직 OS의 적용에 소극적인 행태를 보이고 있다. IED에서 OS의 활용은 늘어가는 data의 처리와 사용자에게 보이는 HMI를 더 효과적으로 관리 할 수 있게 하며, 각종 모니터링과 다른 시스템으로의 이식과 프로그램의 유지, 보수에 빠르게 대처할 수 있게 해준다.

기존의 OS가 적용되지 않는 IED들은 인터럽트를 이용하여 프로그램머가 의도한 실시간 제어가 가능하고 접근이 쉽다는 장점이 있다. 하지만, 다른 Hardware 시스템으로 같은 기능을 구현하기 위해 다시 firmware 작업을 해야 한다는 부담감이 생기며, 많은 양의 data를 처리하기 위해서 효율적인 firmware 구성이 쉽지 않은 것이 사실이다.

이러한 문제를 해결하기 위해 최근 들어 OS의 사용에 대한 필요성이 부각되어 적용되기 시작하였으며, 그에 필요한 OS 적용을 위한 software 설계에 대하여 살펴보고자 한다.

2.1 IED의 기능

IED의 기본적인 기능은 data를 수집하여 그 data에 따라 알맞은 기능을 수행하는 것이다. 취득되는 data는 경우에 따라서 A/D sampling된 값일 수도 있고, 통신을 통하여 입력된 구조화된 data 일수도 있다. 이러한 data를 이용하여 사용자에게 의해 정해진 동작을 수행하게 된다.

2.1.1 알고리즘 연산 수행

IED의 기능 중 가장 기본적인 기능이며, 가장 중요한 기능이라 할 수 있다. 또한, 연산 알고리즘에 따라서 많은 연산 사이클을 소비할 수도 있으며, 일정 시간 안에 연산이 수행되어야만 하는 경우가 많다. 대표적으로 수행되는 알고리즘에는 다음과 같은 것들이 있다.

- (1) 기본과 추정(Fourier 변환, 미분방정식 등)
- (2) 실효치 계산(RMS 계산 등)
- (3) 역률, 위상 측정 등

이러한 IED의 알고리즘 측정에 관련된 연산은 특히 연산시간의 주기성에 대한 확립이 중요하다.

2.1.2 HMI 수행

사용자와 device가 직접적으로 만나는 부분으로서 LCD등을 이용한 정보의 표시와 더불어 키 또는 터치패널, 경우에 따라서는 키보드와 마우스를 이용한 입력 또한 처리가 되는 부분이다. 각종 data를 사용자의 요구에 맞게 수정되는 부분도 포함된다.

LCD는 character또는 경우에 따라 graphic형태가 쓰일 수 있으며, 보통 일정시간 간격으로 LCD에 data를 다시 써주어 LCD의 표시가 깨지지 않도록 한다.

키 입력 부분은 인터럽트 또는 일정시간 간격으로 읽어서 입력을 확인하게 되며, 키 입력과 data의 수정간의 매치가 아주 중요한 부분이다.

2.1.3 통신 및 I/O 작동

알고리즘 연산 수행의 결과로 통신 또는 I/O를 통하여 외부에 signal을 줄 수 있다.

통신의 종류로는 RS232와 422, 485통신이 대표적이며, 경우에 따라 시리얼 광통신 또는 이더넷 통신이 쓰이기도 한다. 많이 쓰이는 protocol로는 modbus와 DNP통신이 있으며, 최근 들어 IEC61850의 국제 통신 규격에 맞춰 개발이 진행되고 있다.

I/O 입출력은 사용자측에서 원하는 신호로 세팅하거나 기본적인 입출력을 정하여 각 단자를 지정해 놓을 수 있다. CPU의 GPIO나 주소 분기를 통한 디지털 입출력을 이용한다.

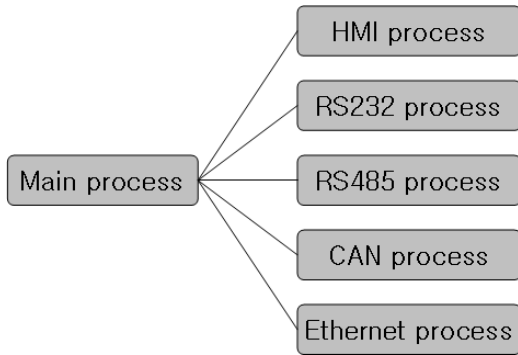
2.2 IED에서 OS의 적용

선택할 수 있는 OS의 종류는 앞에서 살펴본바와 같이 여러 가지가 있다. 본 연구에서는 Open source로서 접근이 용이하고 사용할 수 있는 application이 많이 개발 되어 있는 linux를 이용하였다.

본 연구에서 사용한 linux의 커널은 2.6버전으로 RTOS로서의 동작은 하지 못하지만, 시스템 사이클에 해당하는 시간만큼 process의 cpu 점유를 허락하여 cpu점유를 다른 process에게 넘겨주는 soft RTOS로서 기능이 가능하다.

2.2.1 software 구조 설계

OS가 적용되는 시스템에서는 software를 구성하는 방법에 따라서 시스템의 성능 차이가 나올 수 있다. 아래 그림1은 OS가 적용되는 보드에서 각 process를 구성한 모식도이다.



<그림 1> Process의 구성

전체적인 구성은 Main process를 기준으로 하여 각각의 필요한 process를 생성하는 방식이다. 각 process는 각자가 가지고 있는 기능에 충실 하도록 프로그래밍 되며 process간 data의 교환은 공유메모리와 UDS통신을 이용한다.

IED의 기본 기능 중 알고리즘 연산부분은 Linux의 특성상 연산시간의 주기성이 명확히 확립되기 어려워 다른 CPU를 이용하여 연산되며, 알고리즘 연산 내용을 CAN통신을 이용하여 전달하게 된다. CAN통신을 이용한 알고리즘 연산 내용과 외부 통신 또는 HMI를 통하여 입력된 data는 main process에 전달되며 main process에서는 각각의 다른 process들에게 입력된 data의 내용을 전달한다.

2.2.2 각 부분의 모듈화

OS상에서 동작하는 process에 대하여 각각의 역할에 따라 모듈형태로 구성함으로써 data와 필요 동작의 병렬처리를 가능하게 한다. 모듈화의 가장 큰 장점으로는 모듈별 재사용성이 확립된다는 것이다. software의 큰 수정 없이 같은 OS의 system이면 똑같이 적용 가능하다. 또한, 모듈별 기능이 명확해지기 때문에 유지, 보수에 system전체를 건드릴 일이 거의 없어지며, 각 모듈별 개발자원 할당도 명확히 할 수 있다.

(1) Main process 모듈

Main process는 data관리와 다른 process관리를 하는 manager역할을 한다. 필요에 따라 다른 process를 생성 혹은 삭제하며 각 process들에게 data를 전송하는 역할을 한다.

(2) HMI process 모듈

HMI process 모듈은 LCD의 data display와 키 등으로부터 입력받아 data를 수정, 생성하여 main process에 전달한다. LCD display의 주요 내용은 알고리즘 연산의 내용이 주를 이루게 되며, 기타 필요한 parameter를 사용자에게 보여준다.

(3) RS232/485/Ethernet process 모듈

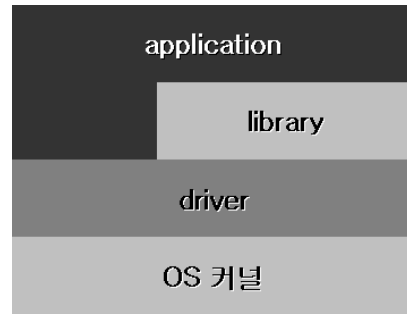
RS232/485 process모듈은 system 외부에서 들어오는 통신 정보를 분석하여 그에 맞는 동작을 수행한다. data의 수정 또는 송신 요청이나 system의 parameter 설정 그리고 system의 I/O 동작 등을 통신으로부터 받아 수행하게 된다.

(4) CAN process 모듈

CAN process 모듈은 IED 내부에서 CPU간의 통신을 책임지며 특히, 연산 알고리즘을 수행한 CPU의 결과를 전송받아 다른 process들에게 전파하는 역할을 한다.

2.2.3 모듈화를 위한 OS기능의 모듈화

OS를 이용한 software를 구성하면 OS에서 제공하는 기능을 사용하여 프로그래밍을 할 수 밖에 없다. OS에서 제공하는 기능은 software와 밀접하게 연관되어 동작하게 된다. 특히 CPU외부 device나 CPU 고유의 peripheral을 사용하게 될 때는 OS에서 지원하는 device driver에 의존하게 된다. 이러한 OS제공 기능과 device driver를 여러 process에서 공통된 interface를 통하여 사용하게 되면 process의 내부에 모듈화가 이루어 질 수 있다. 이런 모듈화의 이점은 process 모듈화와 비슷한 장점을 지니게 되며, 거기에 더하여 OS의 기능에 해당하는 부분만 수정하여 다른 OS에도 쉽게 적용할 수 있는 장점을 가지게 된다. Linux에서 지원하는 기능은 다음과 같이 모듈화를 진행 할 수 있다. 본 개념을 그림으로 도시하면 그림 2와 같다.



<그림 2> OS기능의 모듈화를 위한 구성

OS커널의 기능은 driver계층을 이용하여 application에서 접근할 수 있으며, library계층은 driver를 사용하는데 쓰임이 빈번한 기능을 따로 구성하여 하나의 계층으로 만든 것이다. 다음은 Linux의 기능 중 driver 계층으로 만든 요소들이다.

(1) poll

poll은 linux에서 제공되는 기능으로 process가 필요로 하는 event를 설정하여 event 발생시까지 sleep함으로써 process가 불필요하게 cpu를 사용하는 것을 방지한다.

(2) shm

shared memory로서 각 process가 서로 data를 공유하고 전달하기 위해서 사용한다.

(3) time

linux 시스템의 시간을 가져오거나 설정할 수 있으며, 가져온 시간은 1970년 이후부터 초로 환산된 절대시간이나 연월일로 환산된 시간을 가져올 수 있다.

(4) uds

process에서 shm과 같이 data를 공유하거나 전달하기 위해 사용하는 통신 방법이다.

위에 기술한 driver들 외에 필요에 따라 더 많은 기능의 driver를 생성할 수 있다. 위의 driver들을 자주 사용하는 기능으로 구성하여 application에서 사용하기 편리하게 만든 것이 library이며, 각각의 driver에서 library를 생성하여 사용할 수 있다.

3. 결 론

IED에 OS를 적용시킴으로서 data의 처리와 기능 구현에 모듈화를 진행 하여 성능의 향상과 동시에 유지, 보수의 편리성을 향상시킬 수 있었다. 특히 OS기능의 모듈화를 통하여 각 process의 생성 작업 또한 공통적인 부분을 제외한 나머지 부분만 개발을 담당하는 형태가 되어 개발자원의 유용성을 높일 수 있었다.

하지만 OS적용에 따른 hardware의 가격 상승과 OS자체의 license 가격 등 가격 상승요인이 작용하는 것에 대한 대비가 함께 이루어져야 할 것이다.

[참 고 문 헌]

- [1] 렌 베스 외, “소프트웨어 아키텍처 이론과 실제”, 에이콘, 2007
- [2] Paul Clements외, “Evaluating software Architectures”, Addison-Wesley, 2005
- [3] 최숙영, 문승진, “임베디드 리눅스 기반 산업용 무선 HMI 소프트웨어 모듈 설계 및 구현”, 퍼지 및 지능시스템학회 논문지, Vol. 17 No. 3, p336-342, 2007