JANT 한국항행학회논문지
Journal of Advanced Navigation Technology

항공 우주 해상

J. Adv. Navig. Technol. 28(3): 315-322, Jun. 2024

효율적인 자원 활용을 위한 uC/OS-II 기반의 텔레메트리 PCM 엔코더 설계

Design of uC/OS-II Based Telemetry PCM Encoder for Effective Resource Use

김 건 희[1] · 김 복 기[1*]
단암시스템즈 기술연구소

Geon-hee Kim[1] · Bokki Kim[1*]
R&D Center, DANAM Systems, Gyeonggi-do, 13930, Korea

[요 약]

본 논문에서는 정해진 시간 내에 프레임을 전송해야 하는 텔레메트리 시스템에 적용하기 위한 실시간 운영체제 기반의 PCM 엔코더를 제안한다. 대형 비행체의 경우 각 센서 및 주변장치로부터 많은 상태 정보들을 계측하므로 시스템의 복잡성이 높아지는 추세이다. 또한 계측 데이터가 많아지면서 정해진 시간 내에 프레임을 전송하기 위한 PCM 엔코더의 역할이 중요해지고 있다. 기존 일반적인 엔코더는 규격이 변경되거나, 추가 기능 구현 시 유연성이 떨어지므로 이를 보완하기 위한 설계가 필요하다. 이에 작은 임베디드 소프트웨어에 탑재가 가능한 실시간 운영체제인 uC/OS-II를 적용한 PCM 엔코더 설계를 제안한다. 또한, 타당성을 확인하기 위해 태스크의 실행시간을 측정하는 시뮬레이션을 수행하여 성능을 확인하였다.

[Abstract]

In this paper, we proposes real-time operating system based PCM encoder for telemetry system that must transmit frames within a set time. In the case of large aircraft, the complexity of the system is increasing because a lot of state information is measured from each sensor and peripheral device. In addition, as the amount measurement data increases, the role of PCM encoder to transmit frames within a set time is becoming important. Existing encoder is inflexible when changing specifications or implementing additional features. Therefore, a design is needed to supplement this. We propose a PCM encoder design applying uC/OS-II. In order to confirm the validity, a simulation was performed to measure the execution time of the task to confirm the performance.

Key word : Telemetry system, Pulse code modulation, Real time operating system, uC/OS-II, Field programmable gate array,

https://doi.org/10.12673/jant.2024.28.3.315

Received 16 May 2024; **Revised** 14 June 2024
Accepted (Publication) 17 June 2024 (30 June 2024)

***Corresponding Author : Bokki Kim**

Tel: +82-31-538-6008
E-mail: bokki@danam.co.kr

Ⅰ. 서 론

텔레메트리 시스템이란 비행 중에 발생한 이벤트에 대한 상태정보를 센서를 통해 계측하고 비행체의 상태와 비행 궤도 등을 실시간으로 모니터링하고 분석하기 위해 지상에 있는 수신소로 전송하는 시스템으로 항공/우주 및 국방 산업의 비행체 개발에 있어 중요한 역할을 한다. 텔레메트리 시스템을 구성하는 송신부와 수신부 중 송신부에 해당하는 PCM (pulse code modulation) 엔코더는 비행체의 성능을 검증하기 위해 주변 장치로부터 비행정보를 획득하고 지상의 수신소에서 분석이 가능한 IRIG (inter range instrumentation group) -106 표준의 프레임 형식으로 구성하여 전송하는 기능을 담당한다.

항공/우주 및 국방산업이 발전하면서 각 목적에 따라 우주발사체나 탄도 미사일 등의 비행체 개발이 매우 활발하게 진행되고 있다. 발사체 규모가 확대되면서 대형 비행체들은 분산 위치에 있는 센서 및 주변 장치로부터 더 많은 상태정보들을 계측하게 되고 이에 따라 시스템의 복잡성이 점점 높아지는 추세이다[1]. 또한, 계측하는 데이터양이 많아지면서 정해진 시간 내에 구성을 완료하여 전송하기까지 시간 관리가 필요하다. 때문에 PCM 엔코더는 정확한 시간 내에 각각의 센서에서 계측한 상태정보를 획득하고 이를 프레임 형태로 구성하여 전송하는 역할이 중요해지고 있다.

일반적으로 PCM 엔코더는 VHDL (very high speed integrated circuit hardware description language) 로직으로 구현되어 ROM 테이블 정보를 읽어와 프레임을 구성하는 Commutation 구조를 가진다[2]-[5]. 또는 VHDL 로직과 NIOS 프로세서에 C언어로 구현되어 계측된 신호를 수신하고 프레임 구성 및 전송하는 기능을 순차적으로 수행하는 형태의 엔코더 개발도 이루어지고 있다. 이러한 일반적인 엔코더 구조들은 순차적 구성의 특성 상 개발자가 타이밍 구성이 가능하지만 개발 당시에 매번 기능들의 동작 시간을 측정하여 타이밍 최적화를 수행해하는 번거로움이 존재한다. 또한, 요구 규격이 변경되어 측정 하드웨어 보드 수량이나 계측 채널 증가, 통신 길이가 늘어나면 타이밍 최적화를 재 수행해야하는 유연성이 떨어지는 단점을 가진다.

이러한 문제점을 보완하기 위해 텔레메트리 시스템에 실시간 운영체제 (RTOS; real time operating system) 를 적용하는 방법을 제안한다. 실시간 운영체제는 주어진 입력 조건을 정해진 시간 내에 처리하는 운영체제로 운영체제가 정해진 태스크의 우선순위로 스케줄링을 수행한다. 현재 항공/우주 산업이나 의료등 다양한 분야에서 실시간 운영체제를 적용하여 활용하고 있지만, 운영체제의 특성상 다른 임베디드 소프트웨어 보다 메모리 용량이 커야하는 단점 때문에 아직 텔레메트리 시스템의 엔코더에 적용하기 위한 연구는 많지가 않다[6].

하지만 요구 사항이 바뀔 때 마다 동작시간을 측정하여 소프트웨어의 타이밍 최적화를 수행할 수 없고 비행체가 커지고 점점 계측 채널의 수가 증가하면서 텔레메트리 시스템의 신뢰성, 정확성 및 정밀도가 중요하므로 실시간 운영체제를 적용한 엔코더의 연구가 필요하다.

본 논문에서는 다양한 실시간 운영체제 중 uC/OS-II 기반의 PCM 엔코더 설계를 제안한다. uC/OS-II는 작은 임베디드 시스템에 탑재가 가능하며 높은 성능의 시스템에 사용이 가능하고 소스코드 수정 및 커널의 내부 구조를 이해하기 용이하다. 태스크 스케줄링, 공유자원에 대한 접근 제어 등의 서비스가 제공되며 인텔사의 개발 툴에서 NIOS 프로세서의 uC/OS-II의 커널 및 디바이스 드라이버를 지원하므로 uC/OS-II 기반의 PCM 엔코더를 구현하고 성능을 검증한다.

Ⅱ. 엔코더 구조 설계

본 논문에서는 정해진 시간 내에 안정적으로 실행이 가능하도록 PCM 엔코더에 실시간 운영체제인 uC/OS-II를 적용한 구조를 기술한다.

2-1 로직 구조

FPGA를 이용하여 본 논문에서 제안한 PCM 엔코더의 구조도는 다음 그림 1과 같다

전체적인 로직 구조는 시스템에서 사용하는 클럭을 생성하는 PLL과 RS-422 통신의 SDLC (synchronous data link control) 프로토콜을 처리하는 SDLC 인터페이스 로직, LVDS (low voltage differential signaling) 통신을 처리하는 LVDS 인터페이스 로직, 구성한 프레임을 시리얼로 전송하기 위한 프레임 전송 인터페이스 로직과 uC/OS-II 소프트웨어가 구성되는 NIOS 프로세서로 구성한다. 이때 인터페이스 로직에 버퍼를 구현하여 NIOS 프로세서에서 버퍼로 접근하여 데이터를 쓰고 읽을 수 있도록 구현한다.

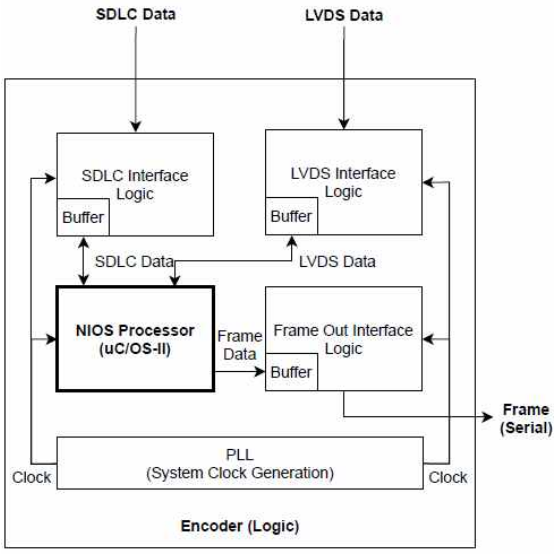

그림1. PCM 엔코더의 로직 구조
Fig. 1. Block diagram of PCM encoder logic.

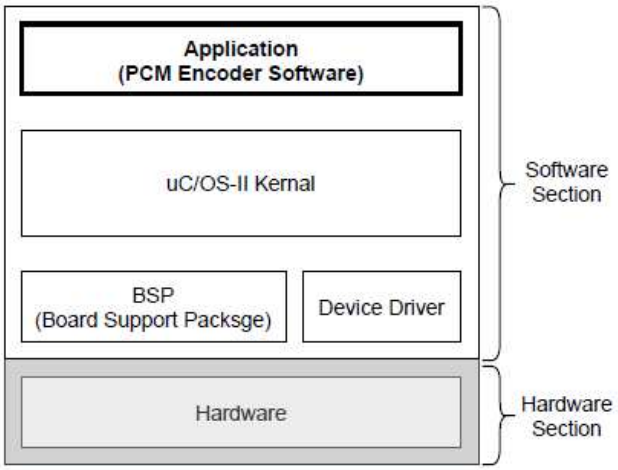

그림 2. uC/OS-II 엔코더 소프트웨어 구조
Fig. 2. Block diagram of uC/OS-II encoder.

2-2 소프트웨어 구조

NIOS 프로세서에서 제공하는 uC/OS-II를 이용한 엔코더 소프트웨어 구조는 위의 그림 2와 같다.

먼저 하드웨어가 구성되고, 하드웨어를 제어하기 위한 디바이스 드라이버와 uC/OS-II 커널로 구성된다. 어플리케이션 영역에는 실시간 운영체제 엔코더 소프트웨어가 구성된다.

PCM 엔코더의 주요 기능은 프레임 구성부, 프레임 전송부 및 데이터 수신부로 구성된다. 엔코더는 200 Hz 주기로 프레임을 구성하여 전송해야 하므로 정해진 5 msec 시간 이내에 모든 기능들이 동작되어야 한다. 때문에 해당 주요 기능을 태스크로 구분하여 그림 3과같이 구현한다.

표 1. 태스크 구성 테이블
Table 1. Table of task.

Task Name	Priority (Task Number)	Execution-Period
Send Task	4	Periodic / Synchronous
Create Task	5	Periodic / Synchronous
LVDS Task	6	Periodic / Synchronous
SDLC Task	7	Aperiodic / Asynchronous

태스크의 구성은 프레임을 구성하는 구성 (Create) 태스크와 구성이 완료된 프레임을 전송하는 전송 (Send) 태스크, LVDS 데이터를 전송 및 수신하는 LVDS 태스크, 마지막으로 수신된 SDLC 데이터를 처리하는 SDLC 태스크로 총 4 개의 태스크가 구성된다. 태스크들과 인터럽트 서비스 루틴들은 이벤트 플래그 및 메시지 큐를 통해 태스크 간의 통신과 태스크의 상태를 대기 및 준비 등으로 제어가 가능하다.

uC/OS-II는 다른 실시간 운영체제와 다르게 우선순위의 중복이 없으므로 태스크 번호가 우선순위를 의미하며 표 1과 같이 구현한 엔코더 소프트웨어의 태스크 우선순위는 전송 태스크, 구성 태스크, LVDS 태스크, SDLC 태스크 순으로 할당한다. SDLC 태스크를 제외한 나머지 태스크들은 외부 인터럽트 신호에 따라 태스크 상태가 제어되므로 주기성을 가진다. 이때 uC/OS-II의 태스크는 최대 64개 중 사용자가 상위 4개와 하위 4개를 제외한 56개의 태스크를 생성가능하며 태스크 번호는 4부터 부여가 가능하다[7].

먼저 우선순위가 최상위인 전송 태스크의 순서도는 다음 그림 4와 같이 구현한다.

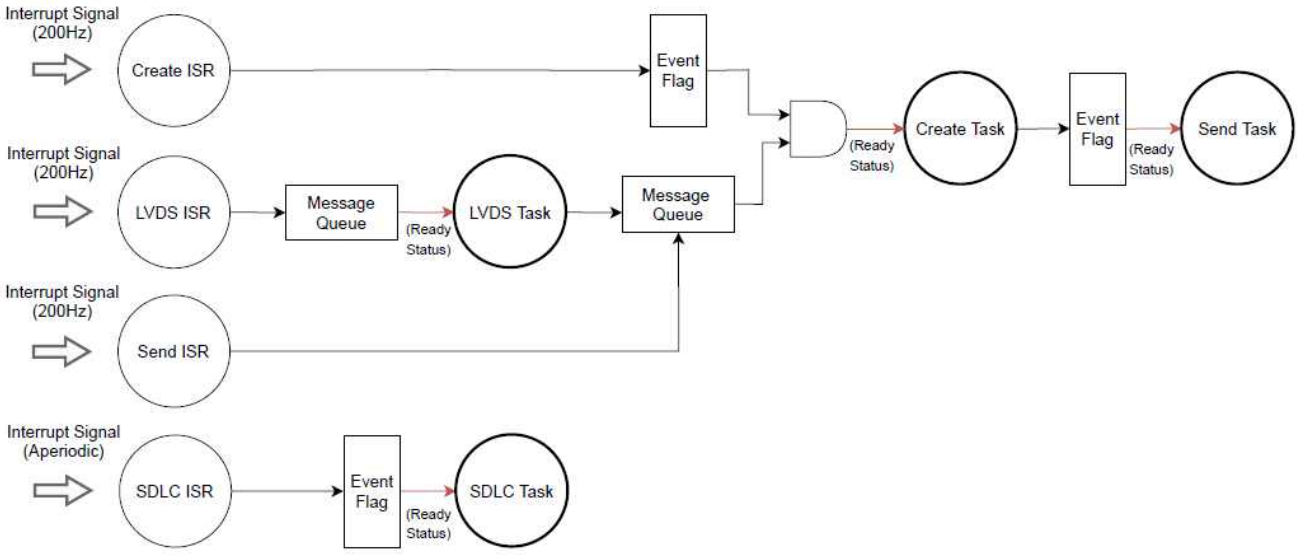

그림 3. PCM 엔코더의 태스크 구조
Fig. 3. Block diagram of PCM encoder task.

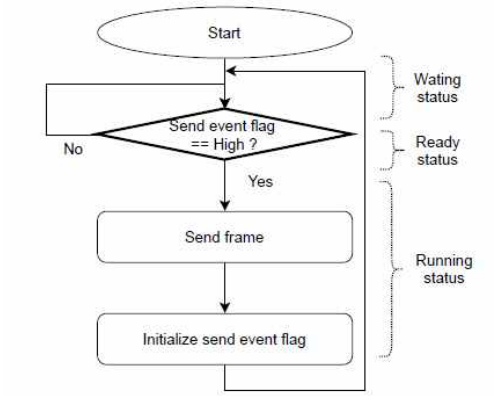

그림 4. 전송 태스크의 순서도
Fig. 4. Flow chart of send task.

전송 태스크는 프레임을 전송하는 태스크로 프레임 구성이 완료될 때까지 대기한 후 프레임 구성이 완료되면 해당 프레임을 프레임 전송 버퍼에 저장하는 역할을 담당한다. 이때 전송 이벤트 플래그를 이용하여 전송 태스크의 상태를 제어한다. 우선순위가 최상위이므로 전송 태스크가 준비 상태가 되면 가장 먼저 CPU (central processing unit)를 선점하여 태스크를 실행시킨다.

아래의 그림 5는 프레임 구성을 담당하는 구성 태스크의 순서도를 나타낸다.

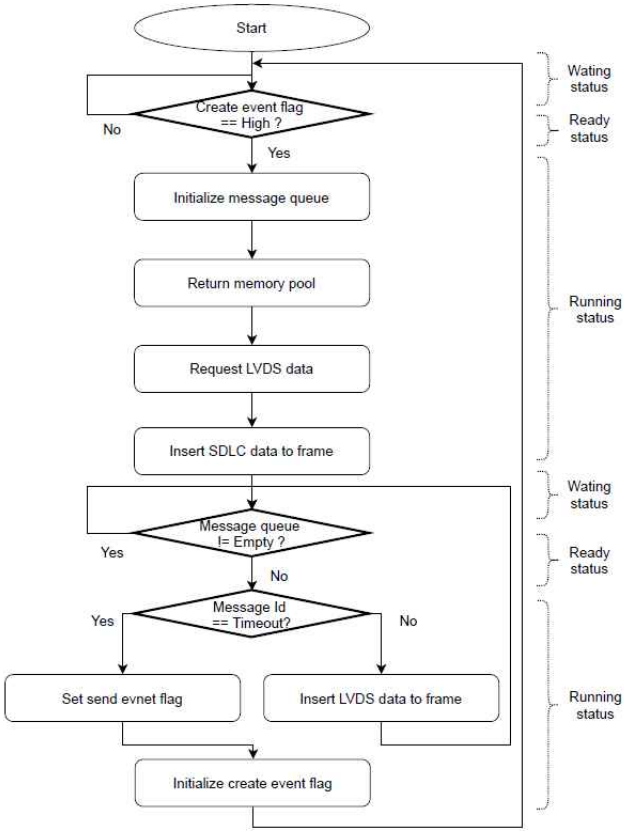

그림 5. 구성 태스크의 순서도
Fig. 5. Flow chart of create task.

구성 태스크는 프레임을 구성하는 태스크로 LVDS 및 SDLC 태스크에서 전송한 데이터를 메시지 큐를 통해 수신하고 모니터링 데이터를 삽입하여 프레임을 구성하는 역할을 수행한다. 구성 태스크는 200 Hz 주기마다 로직 블록에서 발생하는 프레임 구성 이벤트를 수신할 때까지 대기한 후, 구성 이벤트 플래그가 수신되면 프레임을 구성을 시작한다. 이때 메시지 큐를 초기화한 뒤, 메모리 풀을 반환하여 프레임을 구성할 준비를 하고 메시지 큐에 데이터가 전송될 때까지 대기한다. 메시지 큐를 통해 데이터가 수신되면 구성 태스크를 준비상태로 전환하여 태스크를 수행하게 된다. 프레임 구성이 완료되면 전송 이벤트 플래그를 설정하여 전송 태스크의 상태를 준비 상태로 제어한다. 이때 운영체제 커널의 스케줄링을 통해 전송 태스크의 우선순위가 높으므로 CPU를 선점하여 전송 태스크가 수행되게 된다.

다음 아래의 그림 6은 데이터 수신부인 LVDS 태스크의 순서도 이다. LVDS 태스크는 외부에서 LVDS 통신을 통해 데이터를 수신하는 태스크로, 로직 블록에서 발생하는 LVDS 데이터 수신 인터럽트가 발생될 때까지 태스크를 상태를 대기 상태로 유지한다. 인터럽트 서비스 루틴에서 인터럽트 수신 포트 정보를 메시지 큐에 저장하여 LVDS 태스크 상태를 준비 상태로 전환시켜 태스크를 실행한다. 태스크에서 LVDS 데이터 수신 처리가 완료되면 LVDS 데이터를 메시지 큐에 저장하고, 우선순위가 높은 구성 태스크가 CPU를 선점하여 실행될 수 있도록 구성 태스크를 준비 상태로 전환한다.

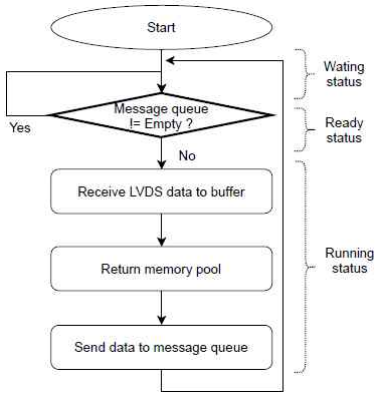

그림 6. LVDS 태스크의 순서도
Fig. 6. Flow chart of LVDS task.

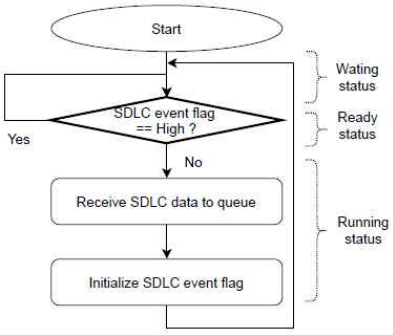

그림 7. SDLC 태스크의 순서도
Fig. 7. Flow chart of SDLC task.

마지막으로 그림 7은 SDLC 태스크의 순서도이다. SDLC 태스크는 비동기로 수신되는 SDLC 데이터를 수신 처리하는 태스크로, 우선순위가 가장 낮으므로 모든 태스크들이 대기 상태일 때 태스크를 수행한다. 외부에서 RS-422 통신을 통해 SDLC 데이터를 수신하면 SDLC 데이터 수신 인터럽트가 발생되고, 인터럽트 핸들러에서 SDLC 태스크를 준비상태로 전환시켜 SDLC 데이터를 수신 처리하도록 구현한다.

이때, 각 태스크를 제어하는 인터럽트는 로직 블록에서 발생하는 신호들이다. 인터럽트 핸들러 내부에서는 인터럽트 서비스 루틴이 끝나기 전 다른 인터럽트 핸들러가 호출되어 태스크의 CPU 선점을 뺏기고, 우선순위가 높은 태스크의 대기 상태가 풀리면서 준비 상태가 된다. 때문에 낮은 우선수위의 태스크는 대기 상태가 되는데 이러한 낮은 우선순위의 태스크들이 중지되는 현상을 방지하고, 인터럽트 핸들러에서 호출한 함수의 내부 동작이 멈추지 않고 정상적으로 수행이 완료되도록 세마포어를 구현한다.

Ⅲ. 시뮬레이션

실시간 운영체제를 적용한 PCM 엔코더의 동작을 검증하기 위해 시뮬레이션을 수행하고 실시간 타이밍을 측정한다.

3-1 CPU 사용률 개선

CPU 사용률은 작업을 수행하는 동안 실제로 사용되는 시간의 비율로 CPU를 얼마나 효율적으로 사용하고 있는지에 대해 소프트웨어의 성능 지표로 사용된다. CPU 사용률을 구하기 위해 총 시간동안의 각 태스크들이 실제로 CPU를 사용하는 실행시간을 측정한다. 시뮬레이션은 엔코더의 유효한 실행시간을 측정하기 위해 200 Hz 주기의 프레임 구성을 총 시간 500 초 동안 10만 번 실행하였고 평균과 분산, 최빈값 및 누적 실행시간을 이용하여 성능을 평가한다.

먼저 기존 일반적인 엔코더의 경우 하나의 main문안에 무한루프로 구현되어 전원이 꺼질 때까지 동작하는 소프트웨어 구조이다. 때문에 while 루프를 돌면서 함수의 동작을 수행하므로 CPU를 계속적으로 점유하여 사용한다. 이때 일반적인 엔코더의 실행시간은 while 루프 내에서 프레임 구성 인터럽트를 수신한 후 프레임을 구성 및 전송하고, 프레임 구성 데이터 수신하기까지의 시간으로 측정한다. 측정한 실행시간은 다음 표 2에서 확인이 가능하다.

프레임 구성 데이터 수신은 데이터 수신 처리로 인해 다음 프레임 주기 시작에 영향을 주지 않도록 프레임 주기의 4 msec 이후부터 데이터 수신 처리를 하지 않도록 예외 처리가 되어있다. 그러므로 4 msec 이후에는 while 루프 안에서 새로운 주기가 시작되는 프레임 구성 인터럽트만을 기다리고 있으므로 일정한 상태를 유지한다고 가정한다. 일정한 상태를 유지하는 구간은 대기시간으로 보고 실행시간 측정 구간에서 제외한다.

표 2. 일반적인 엔코더의 실행시간
Table 2. Table of common encoder runtime.

Function	Mean [usec]	Variance	Mode [usec]	Cumulative Execution Time[usec]
main	3997.88	0.21	3998.1	399788747.3

본 논문에서 제안하는 엔코더의 경우 4개의 태스크로 구성되어있고 각각의 태스크들은 우선순위와 신호에 따라 태스크가 수행되므로, 태스크의 대기시간을 제외하고 태스크들이 실제 CPU를 사용하여 실행하는 시간을 측정하였다. 측정한 각 태스크별 실행시간에 대한 데이터는 다음의 표3과 그림 8과 같다.

전송 태스크의 실행시간은 구성 태스크에서 프레임 구성이 완료된 후 전송하는 이벤트 신호에 따라 실행된다. 때문에 구성 태스크의 신호를 수신한 후부터 전송 태스크에서 프레임을 전송하기까지의 태스크 실행시간을 측정한다.

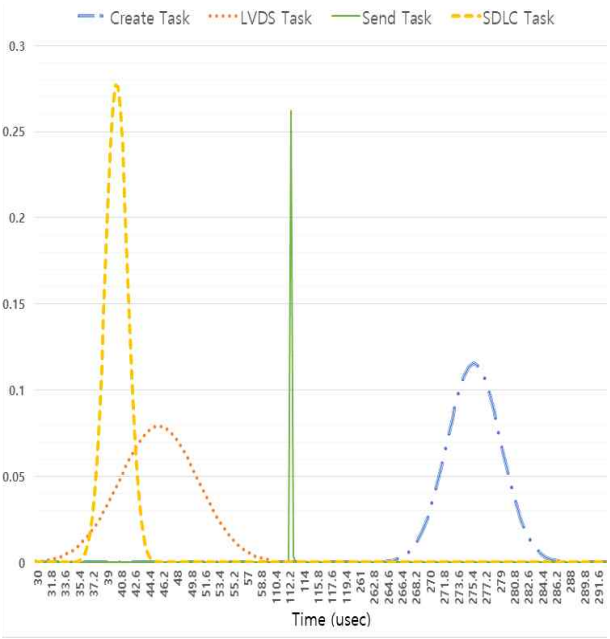

그림 8. 태스크 실행시간의 정규분포
Fig. 8. Normal distribution of task execution time.

표 3. uC/OS-II 엔코더의 실행시간
Table 3. Table of uC/OS-II encoder task execution time.

Task Name	Mean [usec]	Variance	Mode [usec]	Cumulative Execution Time[usec]
Send Task	112.62	0.002	112.6	11262028.5
Create Task	275.78	11.92	277.4	27578232.5
LVDS Task	45.74	25.38	47.9	4574387.8
SDLC Task	40.33	2.06	40	4032756.4

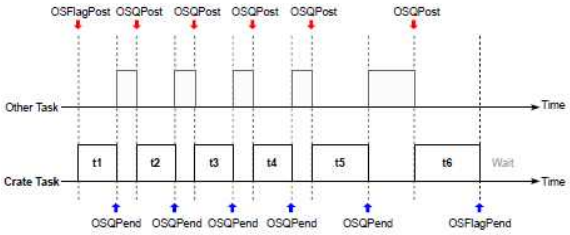

그림 9. 구성 태스크의 실행시간 측정 타이밍도
Fig. 9. Timing diagram of create task.

구성 태스크는 로직에서 발생하는 프레임 구성 이벤트 기준으로 구성을 시작하며 메시지 큐에 데이터 수신에 따라 태스크가 실행된다. 때문에 구성 이벤트 플래그 및 메시지 큐 데이터 수신 대기를 제외한 실행시간을 위의 그림 9와 같이 측정한다.

t1의 시간은 프레임 구성 이벤트 플래그를 수신한 후부터 구성을 시작하여 메시지 큐에 데이터 수신 대기 전까지의 실행시간이다. t2~t5는 LVDS 태스크로부터 메시지 큐 데이터가 수신되고 이를 처리하기까지의 실행시간이며, t6는 로직으로부터 프레임 전송 메시지를 수신하여 전송 태스크로 프레임 전송 이벤트 플래그를 발생시키기까지의 실행시간을 측정한다.

LVDS 태스크의 경우 LVDS 인터럽트 수신 포트만큼의 실행시간이 측정된다. 현재 구현중인 엔코더의 경우 LVDS 인터럽트 수신 포트는 4개이며 먼저 수신된 포트의 인터럽트부터 수신하므로, LVDS 태스크의 실행시간은 프레임 주기 내의 실행시간을 합한 값을 이용한다.

마지막으로 SLDC 태스크의 경우 프레임 구성 주기와 관련없이 비주기적 및 비동기적으로 동작하는 태스크이지만, 시뮬레이션을 위해 일반적인 엔코더와 동일하게 주기적으로 데이터를 전송하여 성능을 확인한다. 이에 SDLC 태스크가 실행될 때마다 시간을 측정하여 총 시간동안 누적된 실행시간으로 CPU 사용률을 구한다.

CPU 사용률을 구하는 수식은 다음 수식 1과 같다.

$$CPU\,(\%) = \left(\frac{\sum Texe}{Ttotal}\right) \times 100 \quad (1)$$

Texe는 태스크들의 실행시간 합이며 Ttotal은 총 시간이다.

10만 번이 수행된 총 시간은 500 초로 전체 실행시간을 측정하여 구한 CPU 사용률은 기존 일반적인 엔코더의 경우 누적된 전체 실행시간이 399788747.3 us 로 약 79.96%를 사용한다.

uC/OS-II 엔코더의 경우 4개의 태스크의 누적된 전체 실행시간이 47447124.1 us이므로 CPU를 9.49%만큼 사용한다.

실제로 임베디드 소프트웨어에서는 CPU 사용률을 70% 이하로 유지하는 것이 바람직하다. 기존 일반적인 엔코더의 경우 70%가 넘는 수치이므로 인터럽트 등의 처리가 추가되면 오작동할 가능성이 매우 높다. 하지만 제안한 uC/OS-II 엔코더의 경우 태스크들이 대기할 동안은 CPU를 사용하지 않으므로 10% 이하의 사용률을 보인다. 때문에 인터럽트가 추가되거나 태스크의 동작이 추가되더라도 소프트웨어가 오작동할 가능성이 낮고, 기존 일반적인 엔코더보다 CPU를 효율적으로 사용하고 있어 시스템의 안정성 및 신뢰성이 향상된 것을 확인 가능하다.

3-2 소프트웨어 복잡도 개선

기존 일반적인 엔코더 소프트웨어는 전체적인 시간을 측정하여 최적화 시간으로 타임아웃을 설정하여 동작한다. 때문에 기능이 추가되면 타이밍 도를 새로 그리고 최적화 시간을 측정하여 반영해야하는 타이밍 재설정 등 구현의 복잡성이 존재한다. 또한 인터럽트 수신 시간에 따라 처리하는 루틴에 영향을 주기 때문에 수신 타임아웃을 설정하여 예외 처리 하도록 구현되어있다.

본 논문에서 제안한 엔코더의 경우 정해진 우선순위에 따라 운영체제 커널에서 스케줄링 해주므로 타이밍 재설정 등의 추가 작업을 하지 않아도 된다.

그림 10의 두 번째 파형과 같이 비주기적으로 수신되는 SDLC 데이터를 기존 일반적인 엔코더의 경우 프레임 구성되는 동안 처리 하지 못하고, 전송이 완료된 이후부터 데이터 수신 처리가 이루어진다. 그리고 다음 프레임 주기의 영향을 주지 않도록 수신을 하지 않도록 예외 처리가 된 구간이 존재한다.

반면에 uC/OS-II 엔코더의 경우 그림 11처럼 SDLC 데이터를 태스크의 우선순위에 따라 높은 태스크들부터 동작하고, 가장 우선순위가 낮은 SDLC 태스크가 실행되도록 스케줄링이 되고 있음을 확인 가능하다.

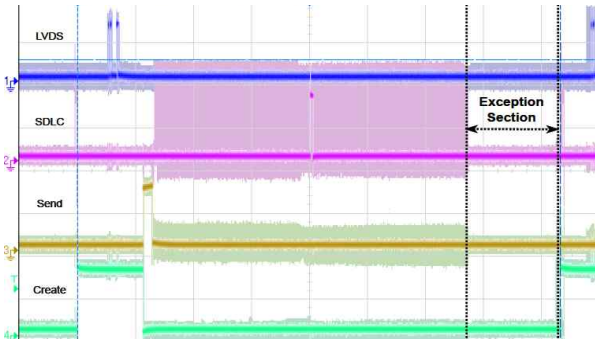

그림 10. main 루프의 SDLC 데이터 처리 파형
Fig. 10. SDLC data processing waveform of main loop.

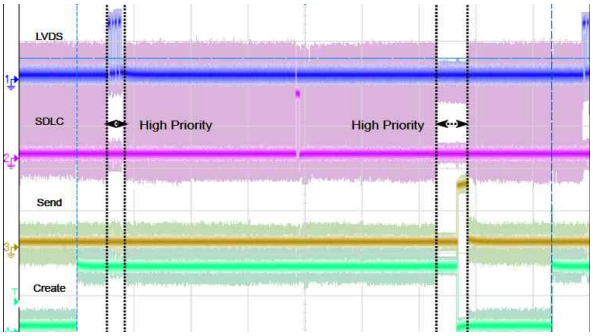

그림 11. SDLC 태스크의 SDLC 데이터 처리 파형
Fig. 11. SDLC data processing waveform of SDLC task.

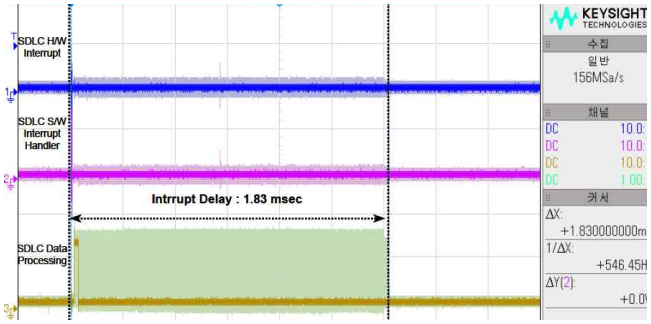

그림 12. main 루프의 인터럽트 처리 지연 시간
Fig. 12. Interrupt processing delay waveform of main loop.

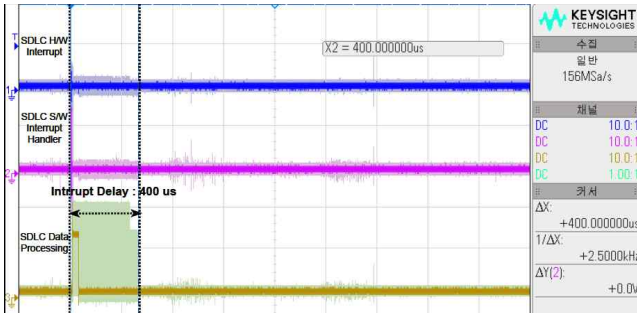

그림 13. SDLC 태스크의 인터럽트 처리 지연 시간
Fig. 13. Interrupt processing delay waveform of SDLC task.

보통 하드웨어 인터럽트가 수신되면 바로 소프트웨어에서는 인터럽트 핸들러가 호출되고 SDLC 수신 데이터 처리가 이루어진다. 기존 일반적인 엔코더의 경우 프레임 구성이 되는 루프 중간이나 예외 처리 구간에 발생한 SDLC 인터럽트는 다음 프레임 주기의 프레임 전송이 완료된 이후에 처리가 되는 워스트 케이스가 존재한다. 그림 12는 하드웨어 인터럽트 발생 시간을 기준으로 인터럽트 핸들러 시간, SDLC 데이터 처리 시간을 1시간 누적한 파형이다. 이때 하드웨어 인터럽트가 발생한 후 SDLC 데이터 처리까지의 지연 시간은 최대 1.8msec 이다.

제안한 uC/OS-II 엔코더의 워스크 케이스는 SDLC 태스크가 우선순위가 높은 구성 태스크 및 전송 태스크를 먼저 수행한 후 처리가 되는 경우이다. 그림 13은 uC/OS-II 엔코더의 누적 파형이며 이때의 SDLC 데이처 처리까지 지연 시간은 최대 400 usec 이다. 측정한 결과를 보면 제안한 엔코더가 기존 일반적인 엔코더에 비해 지연되는 시간이 1.4 msec 이상 작음을 확인 할 수 있다. 이는 일반적인 엔코더는 while 루프를 순차적으로 수행하고 정해진 타이밍에 따라 동작하기 때문이다. 반면에 uC/OS-II 엔코더는 태스크의 우선순위에 따라 스케줄링이 되므로 지연 시간이 적고 보다 효율적으로 구현이 가능하다.

Ⅳ. 결 론

비행체의 성능을 실시간으로 분석하기 위해서는 비행시험 중에 획득한 데이터들을 정해진 시간 내에 전송하여 데이터의 안정성, 신뢰성 및 정확성이 중요하므로 실시간 운영체제를 적용한 엔코더의 연구가 필요하다. 본 논문에서는 실시간 운영체제인 uC/OS-II를 엔코더에 적용하여 4개의 태스크로 구현하고 시뮬레이션을 통해 성능을 검증하였다.

시뮬레이션을 통해 측정한 실행시간을 통해 각 엔코더들의 CPU 사용률 측정하였으며 기존 일반적인 엔코더의 경우 79.96%인 반면, 제안한 엔코더의 경우 9.49%로 효율적으로 CPU를 사용하고 있는 것을 확인할 수 있다. 이를 통해 새로운 기능을 추가 구현하거나 인터럽트 등의 작업이 추가될 경우 기존 일반적인 엔코더는 소프트웨어가 오작동할 확률이 높으므로, 정상적으로 동작하기 위해 전체적인 동작 타이밍을 재 측정해야하는 번거로움이 있다. 하지만 uC/OS-II 엔코더의 경우 타이밍 측정 없이 태스크 우선순위의 따라 동작함으로 보다 효율적으로 개발이 가능하다. 이처럼 텔레메트리 시스템의 PCM 엔코더에 실시간 운용체제를 적용한다면 기존의 일반적인 엔코더보다 효율적으로 CPU 자원을 활용이 가능하고 소프트웨어 개발하는데 있어 복잡도를 개선할 수 있다.

References

[1] S. J. Kim, J. H. Kim and K. S. Kim, “Superposition modulation for high-throughput telemetry communication system,” in *Proceeding of the Koran Institute of Communications and Information Sciences,* Pyeongchang: Korea, pp. 1034-1035, Feb. 2020. Retrieved from https://www.dbpia.co.kr

[2] G. H. Kim, M. H. Jin and B. K. Kim, “Design of a simple PCM encoder architecture based on programmable ROM,” *Journal of Advanced Navigation Technology,* Vol. 23, No.2, pp. 186-193, Apr. 2019. DOI: https://doi.org/10.12673/jant.2019.23.2.186

[3] H. M. Eckstein, “A programmable-signal conditioning pulse code modulated telemetry encoder,” in *Proceedings of the International Telemetry Conference*, San Diego: CA, Vol. 17, pp. 1319-1328, Oct 1981. Retrieved from https://repository.arizona.edu

[4] N. C. Poirier and T. P. Wheeler, “Programmable PCM encoder.” in *Proceedings of the International Telemetry Conference*, Vol. 20, Oct 1984. Retrieved from https://repository.arizona.edu

[5] Y. H. Ro, G. H. Kim, D. Y. Kim, B. K. Kim and N. S. Lee, “Design of advanced PCM encoder architecture for efficient channel information memory management,” *Journal of Advanced Navigation Technology,* Vol. 24, No.4, pp. 305-313, Aug. 2020. DOI: https://doi.org/10.12673/jant.2020.24.4.305

[6] J. W. Yang, H. J. Kim and H. Y. Jang, “A study on the development of RTOS-based engine control system,” in *Proceeding of the Aerospace System Engineering,* Jeju, pp. 820-821, Oct. 2023. Retrieved from https://www.dbpia.co.kr
[7] J. J. Labrosse, *MicroC/OS-II The Real-Time Kernel,* 2nd ed. Florida, FL: CRC Press, 2002.

김 건 희 (Geon-hee Kim)
2014년 2월 : 서울과학기술대학교 전자정보공학과 (공학사)
2015년 4월 ~ 현재 : 단암시스템즈(주) 통신기술연구소 재직 중
※관심분야 : 항공 통신 시스템, 원격측정장치

김 복 기 (Bokki Kim)
1995년 2월 : 서울대학교 수학과 (이학사)
1997년 2월 : 서울대학교 수학과(정수론) (이학석사)
1997년 1월 ~ 2002년4월 : 단암전자통신(주) 연구소
2002년 5월 ~ 현재 : 단암시스템주(주) 통신기술연구소 재직 중
※관심분야 : 무선통신, 채널코딩, 디지털 신호처리 구조