JANT 한국항행학회논문지 Journal of Advanced Navigation Technology

항행 및 항법

J. Adv. Navig. Technol. 25(4): 273-279, Aug. 2021

무인기를 위한 이중화 비행제어컴퓨터의 동기화 설계

Synchronization Method Design of Redundant Flight Control Computer for UAV

이 영 서* · **강 신 우** · **이 희 곤** · **안 태 식**
LIG넥스원 항공드론연구소

Young Seo Lee* · **Shin Woo Kang** · **Hee Gon Lee** · **Tae-Sik Ahn**
Aerospace/Drone R&D Lab, LIG Nex1, Daejeon, 34127, Korea

[요 약]

무인항공기에 적용되는 비행제어컴퓨터는 safety-critical 구성품으로, 내결함성을 확보함으로써 운용의 신뢰성을 높이기 위해 다중화 구조로 설계되고 있다. 이러한 다중화 구조가 적용된 비행제어컴퓨터는 각각의 독립적인 연산/제어 장치가 동일한 시점에 동일한 작업을 수행할 수 있도록 설계되어야 하며, 이를 위해 각 연산/제어 장치 간의 작업 동기화를 위한 동기화 알고리즘이 포함되어야 한다. 본 논문에서는 무인기에 적용되는 이중화 비행제어컴퓨터간의 동기화를 위한 소프트웨어 설계 방법을 제안한다. 제안하는 동기화 방법은 고장률 감소를 위해 최소의 하드웨어 리소스만을 사용하여 동기화할 수 있도록 설계하였고, 동기화에 사용되는 하드웨어 타이머의 동작 방식을 고려하여 설계함으로써 타이머 동작에 따른 동기화 오차를 최소화 할 수 있도록 설계하였다.

[Abstract]

A flight control computer(FLCC) applied to an unmanned aerial vehicle(UAV) is a safety-critical item, and which is designed in a multiple structure to increase the reliability of operation by securing fault tolerance. These FLCC of multiple structure should be designed so that each independent processing/control components can perform the same operation at the same time. And for this reason, a synchronization algorithm for synchronizing the operation between FLCCs should be included in an operational flight program. In this paper, we propose a software design method for synchronization between dual FLCCs applied to UAVs. The proposed synchronization method is designed to synchronize using only the minimum hardware resources to reduce a failure rate. In addition, the proposed synchronization method is designed to minimized synchronization errors due to a timer operation by designing in consideration of operation characteristics of the hardware timer used for the synchronization.

Key word : Flight control computer, Synchronization method, Synchronization algorithm, Redundant system.

https://doi.org/10.12673/jant.2021.25.4.273

Received 23 July 2021; **Revised** 3 August 2021
Accepted (Publication) 24 August 2021 (30 August 2021)

***Corresponding Author Young Seo LEE**

Tel: +82-042-718-3586
E-mail: youngseo.lee@lignex1.com

Ⅰ. 서 론

무인항공기(UAV; unmanned aerial vehicle)에 탑재되는 비행제어컴퓨터(FLCC; flight control computer)는 각종 센서 정보와 제어 명령을 입력받아 항공기의 구동기를 제어하는 장비이다[1]. 이러한 FLCC는 고장 발생 시, 항공기의 안전에 직접적으로 영향을 미치기 때문에 safety-critical 장비로 분류되고, 신뢰성을 향상시키기 위해 fault tolerance 설계 방법을 적용하여 설계되고 있다[1]. fault tolerance를 위한 설계 방법 중 일반적으로 적용되는 설계 방법은 독립된 하드웨어 및 소프트웨어의 다중화 설계이며[2], [3], 다중화 구조로 개발된 FLCC의 비행운용프로그램(OFP; operational flight program)은 독립적인 다중화 장치가 동일 시점에 동일한 작업을 수행할 수 있도록 task frame의 동기화 기능을 포함하고 있어야 한다.

일반적으로 사용되는 이종 장치간의 동기화는 IEEE 1588 PTP(precision time protocol)[4]을 활용하여 구현될 수 있다. 1588 PTP 기반의 동기화 알고리즘은 가장 정밀한 시간을 보유하고 있는 master 장치가 slave 장치에 시간 정보를 공유함으로써 동기화가 이루어지고, 시간 동기화의 결과로 하나의 네트워크에 연결된 모든 노드는 동일한 global time을 갖도록 동기화될 수 있다. 하지만, 정밀한 동기화를 위해 하드웨어적인 global time module을 포함하고 있어야 하며, 1588 PTP 기반 동기화된 global time은 OFP의 스케줄링에 직접적으로 사용될 수 없기 때문에 사용에 제약이 있다. 따라서 다중화 FLCC 간의 task frame 동기화를 위해선 별도의 동기화 알고리즘이 설계되어야 한다.

다중화 FLCC간의 동기화 알고리즘에 대해선 [1], [3]에서 소개되었다. [1]에서는 하나의 LRU(line replaceable unit)내 2채널로 이중화된 FLCC에서 discrete 신호 입출력을 이용한 양방향 이중 동기화 방식의 개념에 대해 소개하고 있으며, 5 μs 이내의 동기화 성능을 제시하고 있다. [3]에서 소개하는 동기화 방법은 하나의 LRU내 3채널로 다중화 된 FLCC에서 discrete 신호 입출력을 이용한 양방향 3중 동기화 방식의 개념에 대해 소개하였고, 초음속 항공기 동기화 요구시간 300 μs 이내 대비 5 μs 이내의 동기화 성능을 제시하고 있다. 이러한 동기화 설계는 사용하는 타이머의 동작 방식에 따라 의도치 않은 타이밍에 frame의 주기 보정이 이루어져 동기화 오차가 발생할 수 있기 때문에 동기화 성능 개선을 위해선 타이머의 동작 방식을 고려하여 정확한 설계가 적용되어야 한다.

본 논문에서는 이중화 FLCC간의 동기화를 구현하기 위해 별도의 discrete 신호를 사용하지 않고, 타 채널과의 통신을 위해 구성되는 CCDL(cross channel data link)을 활용함으로써 고장률을 감소시킬 수 있는 하드웨어 구성을 소개한다. 또한, 타이머의 동작 방식을 고려한 두 가지 소프트웨어 동기화 방법 설계를 제안한다. 제안된 두 가지 동기화 방법은 실 타겟 기반 환경에서 동기화 성능을 확인하였으며, 그 결과를 제시하였다.

Ⅱ. 동기화를 위한 하드웨어 설계

본 논문에서 제안하는 동기화 방법은 이중화된 FLCC간의 frame 시작 시점의 편차를 보정하여 동기화하는 방법으로, 본 장절에서는 동기화를 위해 사용되는 하드웨어 구성을 기술하였다.

2-1 CCDL

이중화된 FLCC간에 동기화를 위해선, 상대 채널의 FLCC에 자기 채널 FLCC의 frame이 특정 시점에 도달했다는 것을 알려줄 수 있는 triggering 수단이 필요하다. 이러한 triggering 수단은 I/O(input/output) 신호 또는 통신 메시지가 될 수 있다.

이중화된 FLCC는 CCDL을 통해 서로 수집된 데이터 및 연산 결과를 공유하도록 설계된다. 따라서 이중화 구조에서 CCDL은 필수적으로 포함되어야할 설계 요소이며, 동기화에 CCDL을 이용할 경우 별도의 I/O 포트 및 배선을 설계하지 않아도 되기 때문에 동기화에 필요한 하드웨어 리소스 및 배선을 감축시킬 수 있다는 장점이 있다. 또한, 별도의 I/O 신호를 사용하지 않고 최소의 지연 시간을 갖도록 설계한 CCDL 기반의 통신 메시지를 사용할 경우, I/O 신호의 추가 설계가 요구되지 않아 고장률을 감소시킬 수 있다. 이러한 장점으로 본 논문에서는 그림 1와 같이 타 채널 FLCC와의 통신을 위해 설계한 CCDL을 활용하여 동기화 방법을 설계하였다.

2-2 I/O controller

FLCC는 그림 1과 같이 메인 연산 장치인 CPU(central processing unit)와 I/O 및 통신 처리를 위한 I/O Controller를 포함한다. I/O Controller는 Discrete I/O, Analog I/O, 직렬 통신 송/

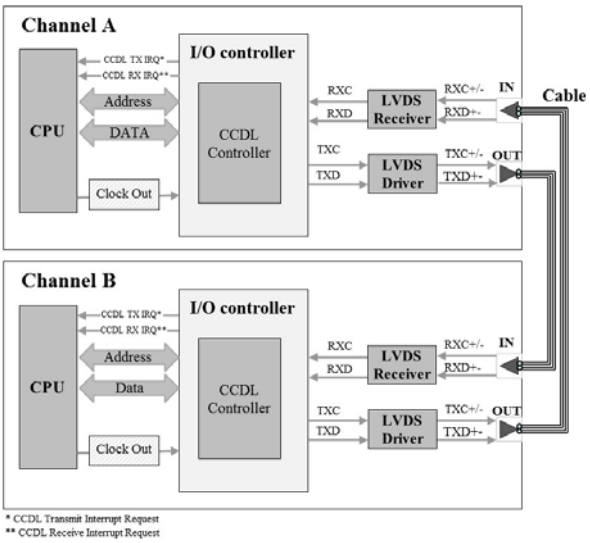


그림 1. FLCC의 동기화 관련 하드웨어 구성
Fig. 1. Hardware configuration for synchronization.

수신 및 CCDL 송/수신을 처리하도록 설계되어 CPU의 I/O 처리를 위한 부하를 절감시키고자 적용된다. 설계한 I/O controller는 I/O 신호 처리 및 통신 송/수신 처리에 대한 지연 시간을 최소화하기 위해 FPGA IP(field programmable gate array intellectual property) core로 설계하였으며, 실시간 병렬처리가 가능하도록 설계함으로써 신호 처리에 대한 지연 시간을 최적화하였다. 또한, 제안하는 동기화 방법 설계를 위해선 CPU는 CCDL 동기화용 데이터(CCDL sync data)의 송/수신 시점을 정확하게 알 수 있어야 한다. 이를 위해 I/O controller는 CCDL sync data 송/수신 완료 시점에 CPU로 인터럽트를 발생시켜 소프트웨어적으로 time-stamp를 기록할 수 있도록 설계하였다.

2-3 Real time interrupt 타이머

일반적인 MCU(micro controller unit) 또는 DSP(digital signal processor)는 real-time interrupt의 구현을 위한 타이머 기능을 제공한다. 이러한 타이머 기능은 base count value register(load count value register 또는 compare counter value register)와 current count value register를 포함하고 있으며, MCU 또는 DSP의 제품군에 따라 up-counter 또는 down-counter를 적용하여 구현된다. up-counter로 동작하는 타이머의 경우 그림 2와 같이 current count value register가 0부터 카운팅을 시작하여 base count value register와 값이 일치했을 경우 인터럽트를 발생시킨다. 반대로 down-counter로 동작하는 타이머의 경우 그림 3과 같이 current count value register가 base count value register에 저장된 값으로 re-load되어 카운팅을 시작하고, 0에 도달했을 때 인터럽트를 발생시킨다.

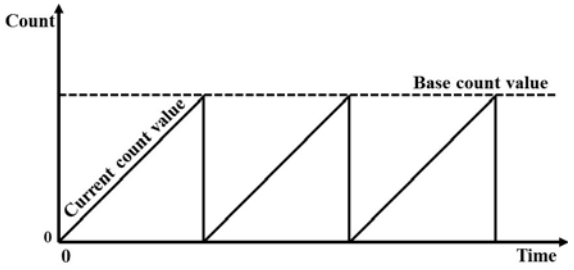


그림 2. Up-counter 기반 타이머의 동작 방식
Fig. 2. Operation of up-counter based timer.

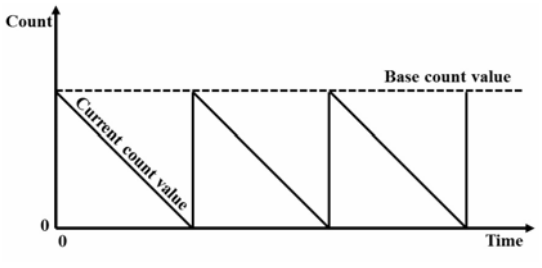


그림 3. Down-counter 기반 타이머의 동작 방식
Fig. 3. Operation of down-counter based timer.

이중화 FLCC간의 frame 동기화를 위해선, 두 FLCC task frame의 시간 편차를 base count value register에 반영함으로써, frame의 주기(다음 frame의 시작 시점)를 조절해야 하며, 사용하는 타이머의 counter 방식에 따라 Frame 주기의 보정 및 보정 주기 반영 타이밍이 달라질 수 있다. 사용하는 타이머가 up-counter라면, 타이머 구동 중에 base count value register를 수정해도 현재 frame의 주기를 즉시 보정 가능하지만, down-counter라면, 타이머 구동 중에 base count value를 수정하더라도 현재 frame의 주기가 즉시 보정되지 않고, current count value register가 보정된 base count value register로 re-load된 시점의 frame 주기가 보정된다. 따라서 down-counter를 사용하는 타이머를 보정하기 위해선 타이머의 base count value register를 보정하는 타이밍과 보정된 값이 실제 적용되는 타이밍을 정확히 설계해야 동기화 오차를 줄일 수 있다.

Ⅲ. 동기화 소프트웨어 설계

본 장절에서는 타이머의 보정 방식에 따른 두 가지 동기화 방법 및 두 가지 동기화 방법에 공통적으로 적용한 설계 고려사항을 기술하였다.

3-1 타이머 보정 방식

Down-counter로 구현된 타이머는 수정한 base count value register를 어느 시점에 task frame의 주기에 적용할 건지에 따라 두 가지 방식으로 보정할 수 있다.

첫 번째 방식은 '보정 값의 현재 주기에 즉시 반영' 하도록 설계하는 것으로, 그림 4와 같이 실행 중인 타이머를 정지하고, base count value register 값을 변경한 후 타이머를 재실행하는 방식이다. 이 방식을 사용하면 계산된 시간 편차의 보정을 수행하는 frame 주기에 즉시 반영할 수 있다. 이 방식을 사용할 때 주의해야할 점은 타이머 정지/재실행 시, 카운팅 된 current count value가 초기 값부터 재 카운팅 된다는 점이다. 따라서 base count value register를 보정할 때, 타이머 정지 시점 까지 카운팅 된 current count register의 값을 반영하여 타이머 재실행 이후에도 이어서 카운팅 될 수 있도록 설계해야 한다.

두 번째 방식은 '타이머 실행 중 반영'하도록 설계하는 것으로, 그림 5와 같이 타이머의 실행 중에 base count value register의 값을 변경하는 방식이다. 이 방식을 사용하면 보정을 수행한 frame의 다음 frame에 보정 값이 반영되어 주기가 조절된다.

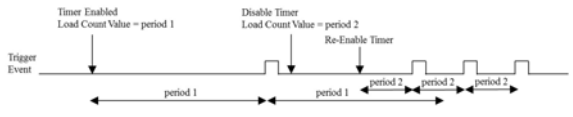


그림 4. 보정 방식 1(보정 값의 현재 주기 즉시 반영)[5]
Fig. 4. Correct mode 1(Modifying Running Timer Period)[5].

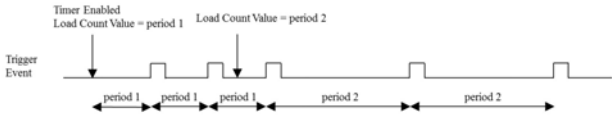


그림 5. 보정 방식 2(타이머 실행 중 반영)[5]
Fig. 5. Correct mode 2(Dynamically Setting a New Load Value)[5]

3-2 동기화 스케줄링 설계

(1) 보정 방식 1 적용 스케줄링 설계: 보정 값을 현재 실행 중인 주기에 즉시 반영하면, 그림 6과 같이 두 frame 당 한번씩 frame의 주기를 보정하도록 동기화 스케줄링을 설계할 수 있다. 매 홀수 frame(2*N*+1 번째)의 시작 시점에 CCDL sync data를 타 채널 FLCC에 전송하고, sync TX(transmit) time-stamp에 현 시점의 누적 카운터 값을 기록하며, 만약 타 채널로부터 CCDL sync data가 수신되었다면, sync RX(receive) time-stamp에 현 시점의 누적 카운터 값을 기록한다. 기록된 sync TX/RX time-stamp는 매 짝수 frame(2*N*+2 번째)에 타 채널 FLCC와의 시간 편차를 계산하는데 활용되며, 계산된 편차에 평균을 취하여 보정 값을 산출한다. 산출된 보정 값은 짝수 frame에서 타이머를 정지한 후 즉시 base count value register에 반영하고, 타이머를 재실행함으로써 짝수 frame의 주기(홀수 frame의 시작 시점 편차)를 즉시 조절할 수 있다.

(2) 보정 방식 2 적용 스케줄링 설계 : 보정 값을 타이머 실행 중 반영하면, 그림 7과 같이 세 frame 당 한번씩 frame의 주기를 보정하도록 동기화 스케줄링을 설계할 수 있다. 매 3번째 frame(3*N*+1 번째)의 시작 시점에 CCDL sync data를 타 채널 FLCC에 전송하고, sync TX time-stamp에 현 시점의 누적 카운터 값을 기록하며, 만약 타 채널로부터 CCDL sync data가 수신되었다면, Sync RX time-stamp에 현 시점의 누적 카운터 값을 기록한다. 기록된 Sync TX/RX time-stamp는 매 3*N*+2 번째 frame에 타 채널 FLCC와의 시간 편차를 계산하는데 활용되며, 계산된 편차에 평균을 취하여 보정 값을 산출한다. 산출된 시간 편차 값은 3*N*+2 번째 frame에서 타이머 동작 중 base count value register에 반영하며, 반영된 base count value는 current count value가 0이 된 이후 re-load 되었을 때 반영되기 때문에 3*N*+3 번째 frame에 적용될 수 있다.

스케줄링 방법 2는 스케줄링 방법 1에 비해 동기화 알고리즘 실행에 대한 시스템의 부하가 낮고, 타이머 중지/재실행에 대한 실행 코드가 미반영 되어 offset이 줄어들 수 있지만, 3*N*+2 frame에서 rate 차이에 의해 누적되는 시간 편차가 동기화에 반영되지 않기 때문에 시간 편차에 더 많은 jitter 성분을 포함 할 수 있다.

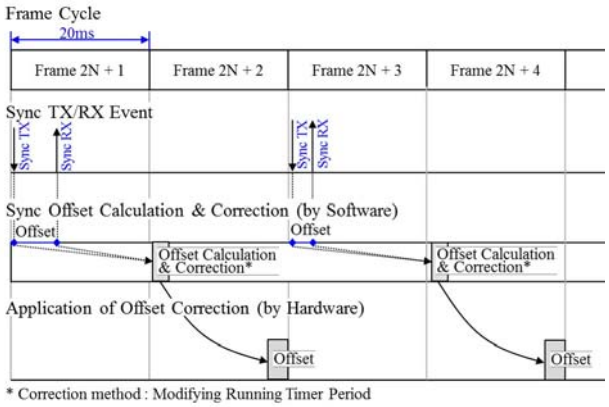


그림 6. 보정 방식 1 적용 시, 동기화 스케줄링 설계
Fig. 6. Synchronization scheduling design 1

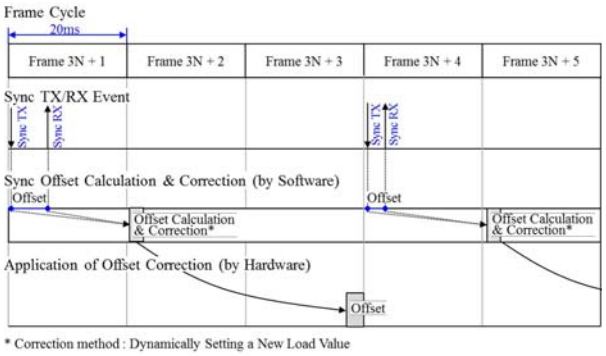


그림 7. 보정 방식 2 적용 시, 동기화 스케줄링 설계
Fig. 7. Synchronization scheduling design 2

3-3 설계 고려사항

본 논문에서 설계한 두 가지 동기화 방법은 공통적으로 소프트웨어 동작의 안정성, 동기화 성능의 최적화를 위해 다음의 4가지 요소에 대해 고려하였다.

(1) 채널 상태 및 CCDL sync data 수신 확인 : 타 채널 FLCC의 정상 상태와 CCDL sync data의 정상 수신 여부를 확인하여 타 채널 FLCC와의 동기화 수행 여부를 결정한다. 타 채널 FLCC의 정상 여부는 채널의 상태가 정상임을 알리는 I/O 신호를 모니터링 함으로써 확인 가능하도록 설계되었다.

(2) 동기화 상태 확인 : 수신된 CCDL sync 데이터에 포함된 타 채널 FLCC의 frame counter 값이 누락 없이 연속된 값으로 수신되고, frame의 시간 편차가 허용 범위(100 μs, 50 Hz의 0.5%) 이내이면 동기화 정상, 그 외의 조건에선 동기화 기능 비정상으로 판단하도록 설계되었다.

(3) 보정 limit value : 동기화를 수행함으로써 frame의 주기가 조절되더라도 각 task의 정상 실행에는 영향을 주지 않아야 한

다. 만약, 보정된 frame의 주기가 한 주기에 실행되는 모든 task의 WCET(worst case execute time) 보다 짧을 경우 task의 실행에 영향을 미칠 수 있기 때문에 시스템에 치명적인 오류가 발생할 수 있다. 따라서, frame 주기를 보정하는 시간 값은 task의 WCET에 영향을 주지 않는 범위로 제한해야 한다. 본 설계에서는 동기화 완료 이전과 이후의 limit value를 각각 400 μs(50 Hz frame 주기의 2%), 4 μs(50 Hz frame 주기의 0.02%)로 설정하여 동기화 동작이 task의 실행에 영향을 주지 않고, 동기화 소요 시간을 줄일 수 있도록 설계하였다.

(4) 동기화 완료 소요 시간 : 동기화 소요시간(*Tsync_complete*)은 초기 부팅 시 각 채널 간에 최대 시간 편차(*△Tmax*), 동기화 알고리즘의 실행 주기(*Tsync_period*), 1 frame에 보정될 수 있는 최대 시간(limit value, *Tlimit*)에 의해 정해지고, 수식 (1)로 나타낼 수 있다.

$$Tsync\_complete = (\triangle Tmax \div Tlimit) \times Tsync\_period \quad (1)$$

스케줄링 방식 1의 경우, 최대 시간 편차(△Tmax) 약 1 frame(20 ms), 동기화 알고리즘의 실행 주기(Tsync_period)가 2 frame(40 ms)이기 때문에 2 s로 계산될 수 있고, 스케줄링 방식 2의 경우 최대 시간 편차(△Tmax) 약 2 frame(40 ms), 동기화 알고리즘의 실행 주기(Tsync_period)가 3 frame(60 ms)이기 때문에 6 s로 계산될 수 있다.

Ⅳ. 동기화 성능 시험 결과

설계한 두 가지 동기화 방법에 대한 안정화 상태에서의 동기화 성능을 확인하기 위해 그림 8과 같이 시험 환경을 구축하였다. 이중화 FLCC는 NXP 사의 MPC5674F를 적용하여 설계하였으며, 스케줄러 제어를 위한 50 Hz의 주기적인 real-time interrupt 생성을 위해 MPC5674F에 내장된 PIT(periodic interrupt time) 모듈을 사용하였다. 설계한 두 가지 동기화 방법은 이중화 FLCC간 PIT 모듈의 타이머를 동기화함으로써 task frame이 동일 시점에 시작되도록 구현하였다. 또한, 동기화 성능 확인을 위해 동기화된 task frame의 시작 지점에서 MPC5674F의 GPIO(general-purpose input/output) 출력을 토글하도록 탐침 코드를 삽입하였고, 두 FLCC에서 출력되는 GPIO 신호를 oscilloscope를 통해 측정하였다. 동기화 알고리즘을 포함한 운용 소프트웨어의 장시간 운용에 따른 최대 동기화 오차를 확인하기 위해 oscilloscope의 측정 파형을 무한히 기록하도록 설정하였으며, 100,000회 누적 기록된 파형 중 최대 편차를 확인하였다.

그림 9는 스케줄링 방법 1을 적용하였을 때의 측정된 동기화 성능이다. 최대 994.44 ns의 offset과 300 ns의 jitter 성분을 포함하고 있음을 확인할 수 있다.

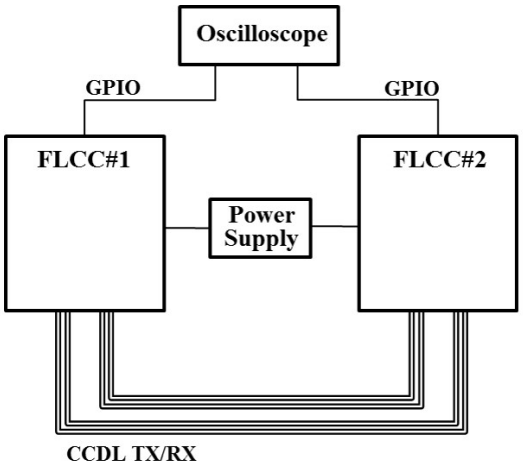


그림 8. 동기화 성능 확인을 위한 시험 환경 구성
Fig. 8. Test environment to verify synchronization performance

그림 10은 스케줄링 방법 2를 적용하였을 때의 측정된 동기화 성능이다. 최대 1.1444 μs의 offset과 555.56 ns의 jitter 성분을 포함하고 있음을 확인할 수 있다. 시험의 결과를 통해 스케줄링 방법 1은 스케줄링 방법 2에 비해 타이머 중지/재실행에 대한 추가 실행 코드가 포함되지만, 동기화 주기가 상대적으로 짧아 rate 차이에 의한 누적 오차가 줄어들기 때문에 동기화 성능의 정밀도가 높아짐을 확인할 수 있다.

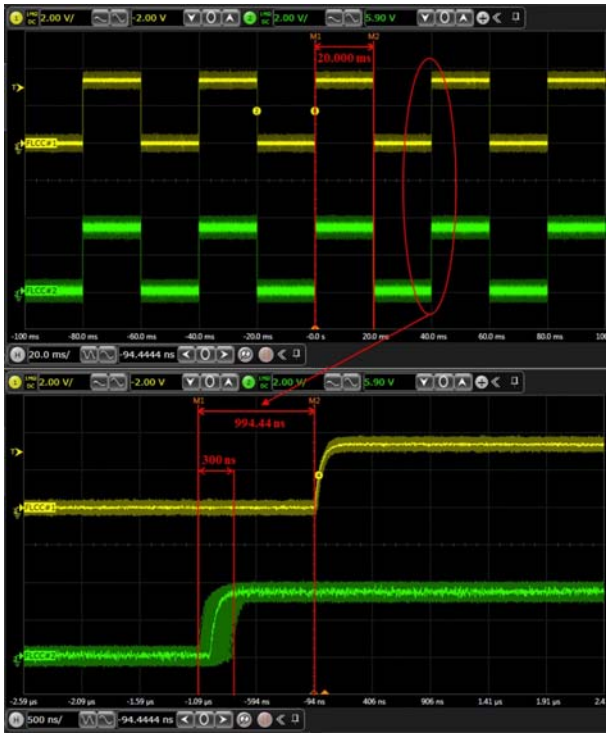


그림 9. 스케줄링 방법 1 적용 시, 시험 결과
Fig. 9. Test result of synchronization method 1

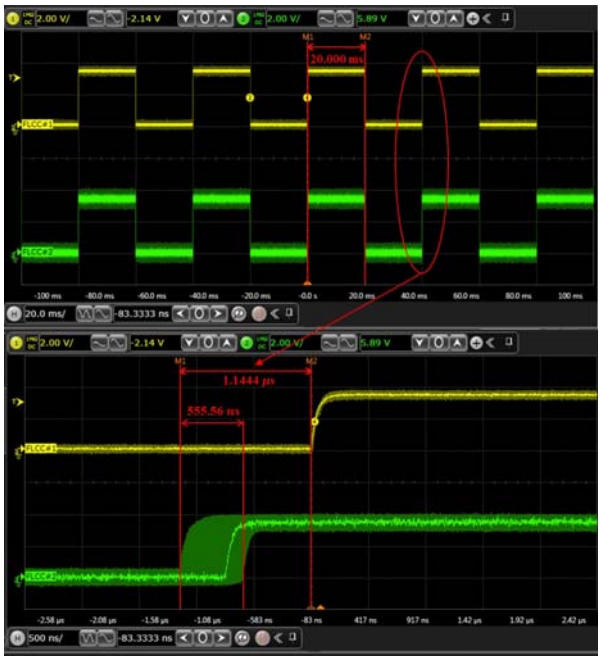


그림 10. 스케줄링 방법 2 적용 시, 시험 결과
Fig. 10. Test result of synchronization method 2

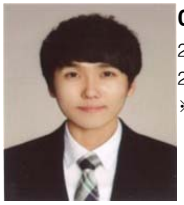
Ⅴ. 결 론

최근 군수분야에서의 UAV 뿐만 아니라, 민수분야에서의 UAM 또는 수송드론 등 다양한 분야에서의 무인항공기 개발이 활발하게 이루어지고 있으며, 다양한 플랫폼의 무인항공기가 실용화되기 위해서는 다중화된 고 신뢰성 비행제어컴퓨터 및 시스템의 적용이 필수적으로 요구된다.

본 논문은 UAV의 이중화 FLCC에 적용할 수 있는 동기화 알고리즘 설계에 대한 논문으로, 시간지연을 최소화한 CCDL 설계를 사용하고, MCU 또는 DSP에 포함되는 real-time interrupt 타이머의 동작 특성을 고려한 두 가지 동기화 설계 방법을 제시하였다. 또한, 제시한 동기화 방법의 기능/성능을 확인하기 위해, MPC5674F MCU 기반의 실 타겟 환경에서 시험하였으며, 이중화 FLCC간의 frame 시작 시점 편차가 시간이 지남에 따라 1.2 μs 이내로 수렴하는 것을 시험 결과를 통해 확인할 수 있었다. 본 논문에서 설계한 동기화 알고리즘은 향후 개발되는 이중화 시스템에 적용되어 간결한 동작에도 정밀하고 안정적인 동기화의 구현을 가능하게 할 것으로 기대한다.

References

1) H. S. Yoon, "Operational Flight Program Development for the UAV Dual Flight Control Computer," in *Korea Computer Congress*, Jeju-do:Korea, pp. 91-93, 2019.
2) S. H. Lee, J. H. Park, C. M. Yum, "Development of Triple Redundancy Flight Control Computer Hardware for UAV," in *Proceeding of The Korean Society for Aeronautical and Space Sciences Spring Conference*, Gangwon-do:Korea, pp. 1021-1025, 2008.
3) D. Y. Choi, B. M. Hwang, I. J. Cho, J. Y. Kim, "A Study on Sync for the Safety-Critical Software of Redundancy System," in *Proceeding of The Korean Society for Aeronautical and Space Sciences Fall Conference*, Gyeongju:Korea, pp. 1021-1025, 2011
4) IEEE(2020, November). 1588-2019 − IEEE Approved Draft Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems [Internet]. Available: https://standards.ieee.org/standard/1588-2019.html.
5) NXP(2015, February). MPC5674F Microcontroller Reference Manual [Internet]. Available: https://www.nxp.com/webapp/sps/download/preDownload.jsp?render=true



이 영 서 (Young Seo Lee)
2016년 02월 : 성균관대학교 전자전기컴퓨터공학과 (공학석사)
2016년 07월 ~ 현재 : LIG넥스원 항공드론연구소 선임연구원
※관심분야 : 임베디드 소프트웨어, 항공전자, 항공기용 비행제어컴퓨터

강 신 우 (Shin Woo Kang)
2002년 8월 : 한국과학기술원 정보공학과 (공학석사)
2002년 10월~현재 : LIG넥스원 항공드론연구소 수석연구원
※관심분야 : 병렬처리, 임베디드 소프트웨어, 항공전자

이 희 곤 (Hee Gon Lee)
2007년 02월 : 원광대학교 전기전자공학부 (공학사)
2020년 01월 : 한국항공우주산업(주) 미래사업부분 위성체계실 선임연구원
2020년 01월 ~ 현재 : LIG넥스원 항공드론연구소 선임연구원
※관심분야 : 항공전자, Reliability test methods development for flight, 항공용 하드웨어 플랫폼

안 태 식 (Tae-Sik Ahn)
2001년 02월 : 경북대학교 전자전기공학 (공학사)
2011년 08월 : 경북대학교 통신공학 (공학석사)
2011년 11년 ~ 현재 : LIG넥스원 항공드론연구소 수석연구원
※관심분야 : 항공전자, 항공기 탑재 컴퓨터, 항공용 비행제어컴퓨터