技術論文 DOI:http://dx.doi.org/10.5139/JKSAS.2011.39.10.979

모델기반 자동코드 생성과 실시간 운영체제 기반 무인기용 비행제어시스템 탑재 프로그램 개발

김성환*, 조상욱*, 김성수*, 유창경*, 최기영**

Development of Embedded Program for UAV Flight Control System using RTOS and Model-Based Auto Code Generation

Sunghwan Kim*, Sang-Ook Cho*, Sung-Su Kim*, Chang-Kyung Ryoo* and Keeyoung Choi**

ABSTRACT

In this paper, an embedded program of a flight control system for a small high performance UAV is introduced. The program consists of modules for device management and guidance and control. The device management system handles navigation sensors and mission equipments. The program for the guidance and control system is used to accomplish various kinds of missions and realize automation of flight control. Driver programs embedded in the device management system for operation of sensors and external devices are based on Texas Instrument's DSP/BIOS RTOS(realtime operating system). The on-board programs for the guidance and control system is obtained by using the model-based auto code generation technology.

초 록

본 논문에서는 소형 무인기용 센서 및 임무장비 통합형 비행제어 시스템의 탑재 프로그램 개발에 대해 다룬다. 비행제어 시스템은 센서 및 임무장비를 관장하는 장치관리시스템과 제어 프로그램 및 임무 수행 알고리즘을 연산하는 제어연산시스템의 두 부분으로 구성된다. 장치관리시스템의 탑재 프로그램은 TI사의 DSP/BIOS 실시간 운영체제를 기반으로 개별 센서 및 외부 임무장비를 위한 시스템 드라이버들과 각각의 서브시스템 관리를 위한 태스크들로 구성된다. 제어연산시스템의 프로그램은 모델기반 개발기술을 이용한 자동 코드 생성 기술을 적용하였다.

Key Words : Model-Based Development(모델기반개발), Auto Code Generation(자동코드생성), Real-Time Operating System(실시간운영체제), UAV(무인항공기), Flight Control Software(비행제어 소프트웨어), DSP/BIOS, RTOS

Ⅰ. 서 론

최근 소형무인기를 이용한 편대임무에 관한 연구가 활발히 수행되었다[1-3]. 이에 따라 소형 무인기의 비행제어시스템은 자동비행 구현의 단순한 수준에서, 다양한 형태의 임무장비를 탑재하고, 다수 무인기간의 협업 시스템 등을 갖춘 센서 및 임무장비 통합형 비행제어시스템으로 발전

† 2011년 5월 18일 접수 ~ 2011년 9월 13일 심사완료
* 정회원, 인하대학교 항공우주공학과
** 정회원, 인하대학교 항공우주공학과
교신저자, E-mail : kchoi@inha.ac.kr
인천시 남구 용현동 인하대학교 항공우주공학과

하고 있다[4-6]. 다양한 서브시스템들을 하나의 통합 시스템으로 구성하는 경우, 임무에 따라 각각의 시스템들을 정확한 시간에 운영할 수 있는 실시간 운영체제 기반의 프로그램이 요구된다.

소형 무인기의 제어알고리즘에 다양한 현대 제어기법[7-8]을 탑재하기 위해서는 복잡한 산술연산이 요구된다. 따라서 탑재 알고리즘 구현에 수동 코딩 개발은 제어 알고리즘을 구현·검증하는 것이 어렵다. 이를 위해 검증된 블록을 이용해 알고리즘 실행코드를 개발·검증하는 모델기반개발(MBD: Model-Based Development) 및 컴포넌트 기반 설계, 객체지향 개발 기법 등이 프로그램 개발에 적용되고 있다[9-10]. 그러나 이러한 자동 코드 생성 기술은 탑재될 하드웨어의 특성을 정확히 파악해야만 실행코드를 실제 시스템에 적용할 수 있다. 때문에 하드웨어 개발자와 제어 알고리즘 개발자가 공동으로 작업해야만 탑재 프로그램이 정상적으로 실행될 수 있고, 이것은 시스템 개발 시간을 지연시키는 요소가 된다.

본 논문에서는 소형 무인기용 통합 비행제어시스템(그림 1)의 탑재 프로그램 개발을 다룬다. 시스템은 크게 센서 및 임무장비를 관리하는 장치관리시스템과 자동비행 제어 알고리즘을 탑재하는 제어연산시스템으로 구성된다. 장치관리시스템의 프로그램은 개별 센서 및 외부 장치의 연결을 위한 시스템 드라이버들과 서브시스템 관리를 위한 태스크들로 구성된 실시간 운영체제가 적용되었다. 제어연산시스템의 프로그램에는 모델기반 자동 코드 생성 기술을 이용하여 제어 알고리즘을 구현하고 실행 프로그램을 생성하여 탑재하였다.

통합 비행제어시스템을 장치관리시스템과 제어연산시스템으로 분리하여 개발하면 하드웨어에 관한 지식이 없는 제어알고리즘 개발자가 하드웨어 개발자의 지원이 없어도 직접 제어연산 시스템에 개발된 알고리즘을 직접 탑재하고 검증할 수 있는 장점이 있다.

그림 1. 통합 비행제어 시스템

II. 본 론

2.1 실시간 운영체제의 필요성

과거 무인기의 자율비행에 초점을 맞춰 개발된 항법용 비행제어시스템의 프로그램은 센서데이터 획득, 항법 및 제어연산, 서보 출력의 일정한 주기 프로그램들의 순차적 배열로 구성된 배경프로세스와 통신 시스템의 데이터 송수신과 같이 시스템 내부·외부 인터럽트에 의해 발생되는 전경 프로세스가 조합된 전경·배경 구조(그림 2)의 프로그램이었다.

전경·배경 구조는 기능이 단순한 프로그램에 대해서는 구현이 쉽다. 그러나 기능이 복잡해지는 경우는, 다수의 기능들이 전경·배경 프로세스에 배치되어, 프로그램들 간에 원활한 데이터 교류가 힘들고, 전경·배경 프로세스 간에 빈번한 스위칭으로 개별 기능들의 실행시간에 대한 정확한 예측이 어렵다. 이것은 다양한 서브시스템으로 구성된 항공기 제어시스템 측면에서 보면, 실시간성이 보장되지 못하므로 항공기 자체의 손실과 직결될 수 있다. 그래서 최근의 비행제어시스템은 이러한 전경·배경 구조를 사용하지 않는다.

전경·배경 구조의 운영 프로그램과는 대조적으로, 실시간 운영체제(그림 3)에서는 여러 기능들

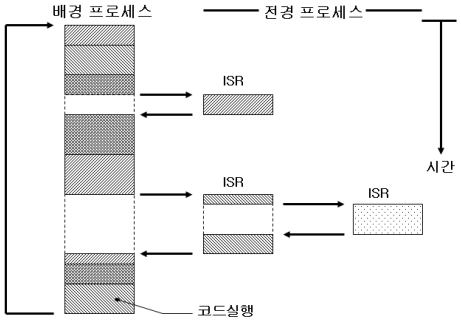

그림 2. 전경·배경 구조의 프로그램 동작

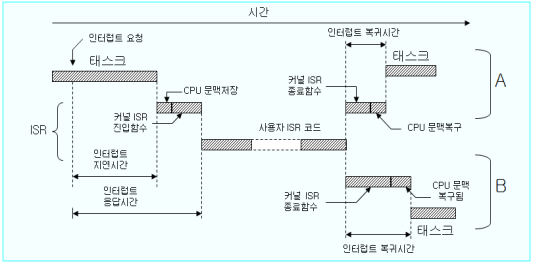

그림 3. 실시간 운영체제에서의 프로그램 동작

이 태스크(Task) 단위로 관리되며, 각 태스크는 중요도에 따라 운선순위를 다르게 적용하고 실행 시작 및 종료 시점의 변경이 가능하다. 이러한 운영체제는 세마포어(Semaphore)나 메시지 큐(Message Queue), 메일박스(Mailbox) 등을 제공해 태스크 간에 데이터 통신 및 동기화를 원활히 수행할 수 있다. 통합형 비행제어시스템은 여러 센서들과 외부 인터페이스를 관리하기 위한 다양한 주변 장치로 구성되며, 각각의 장치들은 기능에 따라 실시간 수행 능력이 매우 중요하다. 이러한 연유로 개발되는 통합형 비행제어시스템은 다양한 기능들을 중요도에 따라 우선순위를 갖는 태스크 단위로 구분하고, 이벤트 발생 즉시 실시간 수행이 중요한 경우는 인터럽트를 활용해 실시간성을 보장할 수 있는 형태로 실시간 운영체제를 구성한다.

2.2 통합 비행제어 시스템 특성

통합 비행제어 시스템(그림 4)은 장치관리시스템과 제어연산시스템으로 구성된다. 통합 시스템은 운영 요구 상황에 따라 장치관리시스템 만을 별도 사용 가능한 형태로 개발되었다. 그림 1에 나타낸 것과 같이 두 개의 시스템으로 구성되는

표 1. 시스템 구성품

프로세서	TMS320F2812, TMS320C6713
센서	자세계, 속도계, 고도계, 받음각, 옆미끄럼각, GPS, RPM, Eng Temp, Board Temp, Battery Volt
통신	RS-232(4채널), CAN, UDP
비행데이터기록	1G x 8비트
입출력	PWM Out(16-ch), Digital IO(8-ch), Analog Input(6-ch)
입력전원	7~12VDC, max. 7W
크기	132(W) x 91(D) x 41(H)mm
무게	410g

데, 위쪽에 배치된 시스템이 제어연산시스템으로 제거가 가능한 형태이다.

비행제어 시스템에 탑재되는 구성품은 표 1과 같다. 장치관리시스템의 프로세서는 TMS320F2812로 다양한 Peripheral을 가지고 있어 많은 서브시스템을 연결하기에 유리한 조건을 가지고 있어 채택되었다. 제어연산시스템은 TMS320C6713 프로세서를 사용한다. 선정 이유는 Floating Point

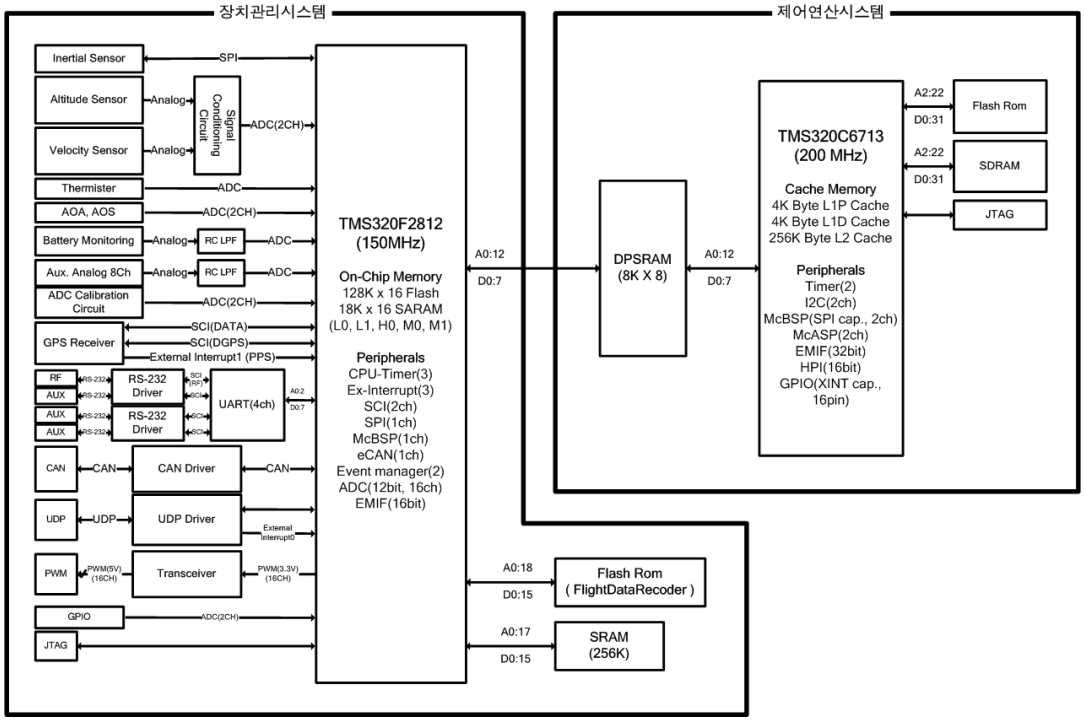

그림 4. 통합 비행제어시스템 개략도

DSP이기 때문에 산술연산이 매우 빠르다. 또한 Matlab Simulink Block에서 지원하는 프로세서로 Auto Code Generation Tool을 이용해 개발된 제어 알고리즘을 중간에 C 코딩의 수작업 과정이 없이 JTAG을 통해 바로 프로그램을 탑재하는 것이 가능하다[4].

2.3 장치관리시스템 프로그램

2.3.1 탑재 프로그램 요구 조건

장치관리시스템에서 탑재 프로그램이 센서 모듈과 주변장치 관리를 위한 요구조건은 다음과 같다.

- 100Hz로 16채널의 Analog 신호를 샘플링 하여 고도, 속도, 받음각, 옆미끄럼각, 시스템온도, 엔진온도, 시스템 전원 등의 데이터 변환
- 100Hz로 IMU로부터 통신 인터페이스를 통해 가속도, 각속도, 자세를 계산
- 100Hz로 8채널의 Digital I/O 제어
- 100Hz로 센서처리 프로세서에서 생성된 데이터를 기록
- 50Hz로 16채널의 PWM 작동기 신호 생성
- GPS의 출력주기에 따라 위치, 속도, 시간, DGPS 보정정보 연산]
- 내·외부 이벤트에 따라 4채널의 RS-232, CAN, LAN 등의 통신 제어
- 100Hz로 장치관리시스템에서 생성된 데이터를 제어연산시스템에 전달

장치관리시스템은 위의 다양한 요구조건에 부합하는 프로그램을 탑재해야 한다. 따라서 여러 기능에 대해 다양한 태스크를 구현하고 실시간 실행을 가능하게 하는 실시간 운영체제를 적용해야 한다.

2.3.2 시스템 드라이버 개발

실시간 운영체제를 지원하기 위한 시스템 드라이버 개발 방법은 크게 폴링(Polling)방식과 인터럽트(Interrupt)방식의 두 가지로 분류될 수 있다. 폴링 방식은 운영체제가 필요에 따라 시스템 드라이버를 호출하여 서브시스템을 제어하는 방법이다. Analog 신호 샘플링과 같은 주변장치 관리 드라이버가 여기에 해당한다. 이 경우 운영체제의 여러 테스크를 관리하는데 있어 드라이버에 의한 시간지연 요소를 예상할 수 있기 때문에 테스크 관리가 쉬워진다. 이에 반해 인터럽트방식은 주변장치에서 CPU에 인터럽트를 발생시키고 인터럽트 처리 구문에서 세마포어나 큐, 메일박스 등을 이용해 드라이버에 이벤트를 전달해주는 방법이다. GPS 데이터 수신과 같이 외부 시스템이 주도적으로 데이터를 송신하는 경우의 드라이버가 여기에 해당된다. 실시간 운영체제에 탑재되는 일부 드라이버의 기능과 지원되는 명령에 대해 아래에 나타내었다.

- SysCtr driver: CPU 내부 장치의 초기화를 위한 드라이버이다. Watchdog, 시스템 clock, Peripheral clock 등의 Register 초기화 함수(InitSysCtrl)와 외부 디지털 입출력관련 Register 초기화 함수(InitGpio), 외부 메모리 인터페이스 속도관련 Register와 메모리 초기화 함수(InitXintf), 내부 플래시 메모리 초기화 함수(InitFlash) 그리고 사용자 초기화 함수(UserInit)를 제공한다.
- UART driver: 내·외부 시리얼 통신 장치 관련 드라이버이다. UART의 Baud Rate과 stop bit, parity bit, 데이터 비트 등을 설정하는 함수(Init_UART)와 UART 관련 메모리 영역의 변수들에 초기화를 위한 함수(UART_Mem_Init), UART에 문제가 발생했을 경우 관련 메모리와 세마포어 등을 초기화 하는 함수(Reset_UART), 특정 UART를 통해 데이터를 전송하는 함수(Write_UART)와 수신하는 함수(Read_UART), UART 관련 인터럽트를 처리하는 함수(UART_TxRx_ISR) 등으로 구성된다.
- DPRAM driver: 장치관리시스템과 제어연산시스템 간에 데이터 통신을 위한 듀얼 포트램을 관리하는 드라이버이다. 메모리 전체 또는 데이터 종류에 따라 구분된 영역의 초기화를 위한 함수(Init_DPRAM), 제어연산시스템에 데이터를 전달하는 함수(DPRAM_Write), 데이터 수신 함수(DPRAM_Read), 인터럽트 처리 함수(DPRAM_ISR) 등의 함수들을 제공한다.

2.3.3 실시간 운영체제(DSP/BIOS) 적용

현재 많은 제어 분야에서 Window, μC-OS, QNX, Embedded Linux등의 다양한 운영 체제가 사용되고 있다[11-13]. 이와 같은 일반적인 범용 운영체제에서는 특정 하드웨어에 탑재하기 위해서는 운영체제의 많은 부분을 수정해야하고 프로세서가 가지고 있는 많은 Peripheral들의 모든 특징을 포괄할 수 없기 때문에 프로세서의 성능을 최대한 끌어낼 수 없다.

TI는 생산 프로세서들에 대해 최적화된 RTOS인 DSP/BIOS를 제공한다. DSP/BIOS의 경우 프로세서에 최적화가 되어 개발된 운영체제이기 때

문에, 다른 운영체제에 비해 프로세서의 성능을 최대한 활용할 수 있을 뿐만 아니라 프로세서에 따라 자체적으로 운영체제의 구조를 맞춰주기 때문에 실행 안정성도 높다.

비행제어 알고리즘의 실행 안정성에 대한 영향을 고려하여, 장치관리시스템의 실시간 운영체제는 각각의 서브시스템들의 기능을 태스크 단위로 관리하며, PWM 생성, 통신포트 제어 등의 하드웨어 기반의 태스크들은 중요도가 높기 때문에 DSP/BIOS 설정에서 우선순위를 높게 설정하였다. Analog 신호 보정, 자세계산 알고리듬 연산 등의 어플리케이션 기반의 태스크들은 상대적으로 우선순위를 낮게 설정하였다. 실시간 운영체제에 탑재되는 일부 태스크의 기능에 대해 아래에 나타내었다.

- IMU TASK: Priority level 12의 TASK로 최초 생성될 때 IMU 장치를 초기화한다. SEM_RUN_IMU Semaphore를 이용 10msec 주기로 IMU의 데이터를 획득한다.
- PWM TASK: Priority level 13의 TASK로 SEM_RUN_PWM 세마포어를 이용 20msec 주기로 제어연산시스템에서 전송된 제어명령을 PWM 신호로 출력한다.
- LAN TASK: Priority level 14의 TASK로 LAN Driver와 연동되어 LAN_TxRx_ISR에서 제공되는 SEM_LAN_RX와 SEM_LAN_TX의 세마포어의 값에 따라 필요시 작동한다. 주로 PILS와 FDR(Flight Data Recoder) 데이터 다운로드에 사용된다.
- UART TASK: Priority level 14의 TASK로 UART Driver와 연동되어 작동하며 4채널에 대해 각각 Rx, Tx TASK들로 구성되며 주로 외무 임무장비 Interface에 사용된다.

표 2는 장치관리시스템에서 수행해야 할 태스크들에 대하여 우선순위를 정리한 것으로 15가 최고 높은 우선순위를 의미한다. 그림 5는 이러한 태스크 우선순의를 Code Composer Studio의 DSP/BIOS Configuration GUI에서 설정하는 부분이다.

실시간 운영체제의 경우 여러 태스크간의 공유데이터에 대한 동시 접근이 문제가 되므로 각 공유데이터에 대하여 충돌 방지를 위한 세마포어의 설정이 요구된다. 그림 6은 이러한 세마포어의 설정을 나타낸다.

실시간 운영체제의 태스크 관리에서 가장 흔히 발생할 수 있는 문제 중 하나는 태스크 관련 스택(Stack)들의 오버플로(Overflow)다. 태스크 스택은 현재 수행중인 태스크와 다른 태스크 간에

표 2. 장치관리시스템 태스크 우선순위 정의

우선순위	기능
15	내부 SCI 통신포트 제어
14	LAN 통신포트 제어 UART(4채널) 통신포트 제어
13	ADC 신호처리, PWM 신호 생성, GPS 데이터 처리
12	IMU 센서와 인터페이스를 통한 데이터 송수신, 비행데이터 기록
11	제어연산시스템에 센서데이터 전송

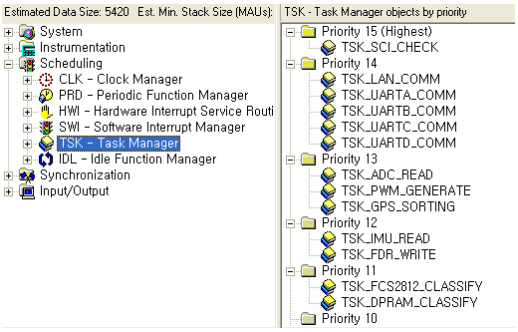

그림 5. DSP/BIOS 태스크 우선순위 설정

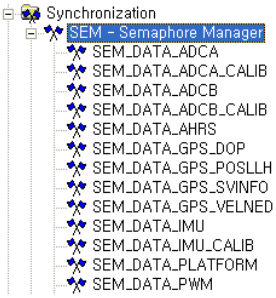

그림 6. 세마포어 설정

스위칭이 발생하는 경우 이전 태스크에서 사용되던 변수나 프로그램 위치 등의 백업 데이터를 저장하기 위한 메모리 영역이다. 스택 메모리가 충분한 크기를 갖지 않는 경우 다른 태스크의 메모리 영역에 데이터를 백업하게 되어 전체 프로그램의 오동작을 일으킬 수 있다. 이를 방지하기 위해서는 각 태스크마다 충분한 크기의 스택을 할당해야한다. Code Composer Studio의 실시간 모니터링 기능을 이용하면 운영체제를 실행하는 동안 각 태스크들이 현재 사용하고 있는 스택의 크기를 분석 가능하다. 이러한 기능을 통해 개발자는 각 태스크들에 대해 최적화된 스택의 크기

를 결정할 수 있다.

그림 7은 스택 오버플로우에 대한 경고화면으로 특정 태스크의 피크치가 사전에 설정된 0x400까지 진행하였기에 경고를 나타내고 있다.

2.3.4 실시간 운영체제(DSP/BIOS) 검증

장치관리시스템의 운영체제에 정상 기능을 확인하기 위하여 기본 기능들의 최대 부하상태에서 프로세서가 정상적으로 동작하는지를 확인하였다. 기본적으로 일정 주기마다 자세계산 및 PWM 생성 등의 기본 기능을 수행함과 동시에 운영 프로그램에 최대 부하를 주기 위하여 외부 임무장비와 통신하기 위한 4채널의 UART 30msec마다 115200 Baud Rate으로 256byte를 송수신하고, PILS 연동을 위한 LAN에 30msec마다 256bytes를 송수신하는 시험 환경을 구현하였다. 그림 8은 이러한 시험환경의 구성이다.

시험은 2번에 걸쳐 2시간씩 시험되었다. 표 3는 위의 환경에서 시험을 했을 때, 데이터의 손실률을 정리한 것이다.

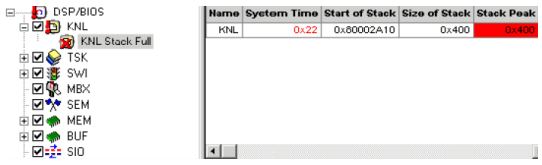

그림 7. 실시간 스택 모니터링

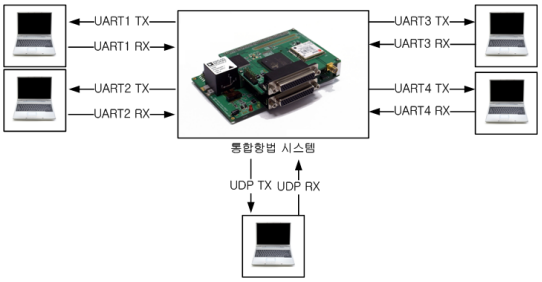

그림 8. 실시간 운영체제 검증 환경

표 3. 최대부하 시험결과

포트	1번 시험의 패킷 손실률	2번 시험의 패킷 손실률
UART 1채널	0%	0.0037%
UART 2채널	0.0046%	0.0039%
UART 3채널	0.0091%	0.0087%
UART 4채널	0.003%	0.0037%
LAN	0%	0%

위의 결과를 살펴볼 때 RS-232 통신방식을 사용하는 UART포트들은 0.01% 미만의 데이터 손실률을 보였다. 이 수치는 232 통신방식의 특성상 장시간 테스트를 수행하는 과정에서 패킷 중에 1~2bits를 잘못 인식하여 데이터 손실이 발생하는 경우 패킷 자체가 손실되기 때문에 발생할 수 있는 상황이다. 이에 반대 LAN 통신의 경우 장시간 테스트를 수행하면서 패킷의 손실이 발생하지 않았다. 테스트 결과 실시간 운영체제를 적용한 장치관리시스템의 운영 프로그램은 임무장비 통신 장치들의 최대 부하 상태에서도 정상적으로 동작하는 것을 확인할 수 있었다.

2.4 제어연산시스템 프로그램

제어연산시스템 프로그램은 주변장치가 DPRAM만을 가지고 있기 때문에 주변장치의 특성이 프로그램 개발에 주요 관심사가 아니다. 대신에 제어 알고리즘 개발자의 편의에 초점을 맞춰, 모델기반 설계기법을 이용해 독립적으로 개발된다.

개발된 제어연산시스템은 높은 계산 능력을 위해 TMS320C6713 DSP 프로세서가 적용되었다. 그림 4의 개략도와 같이 제어연산시스템은 DPRAM을 통해 장치관리시스템과 데이터를 교환한다. 그러므로 Simulink로 구현된 알고리즘을 시스템에 탑재 및 실행하기 위해서는 Simulink 블록 내에서 TMS320C6713 프로세서와 DPRAM을 운용하기 위한 드라이버 관련 프로그램이 필요하다. 이러한 프로그램은 Simulink 환경에서 Custom Code 블록을 이용해 구현될 수 있다.

제어연산시스템의 탑재 프로그램은 그림 9와 같이 크게 3 부분으로 구성되며 네모박스에 포함되지 않는 부분은 전체 프로그램에서 사용되는 여러 공용 변수들의 선언과 데이터 초기화부분이다. 제어연산 시스템의 프로그램은 다음과 같은 기능의 모듈들로 구성된다.

2.4.1 Module 1

알고리즘이 탑재되는 제어연산시스템의 TMS320C6713 프로세서에 관한 설정 부분으로 Simulink Library에서 사용할 하드웨어에 적합한 블록을 선택한 후, Processor 선택, CPU clock, Memory등을 설정할 수 있다. 또한 Section Option을 통해 Compiler에서 요구되는 설정도 가능하다.

2.4.2 Module 2

제어연산시스템의 프로그램은 장치관리시스템

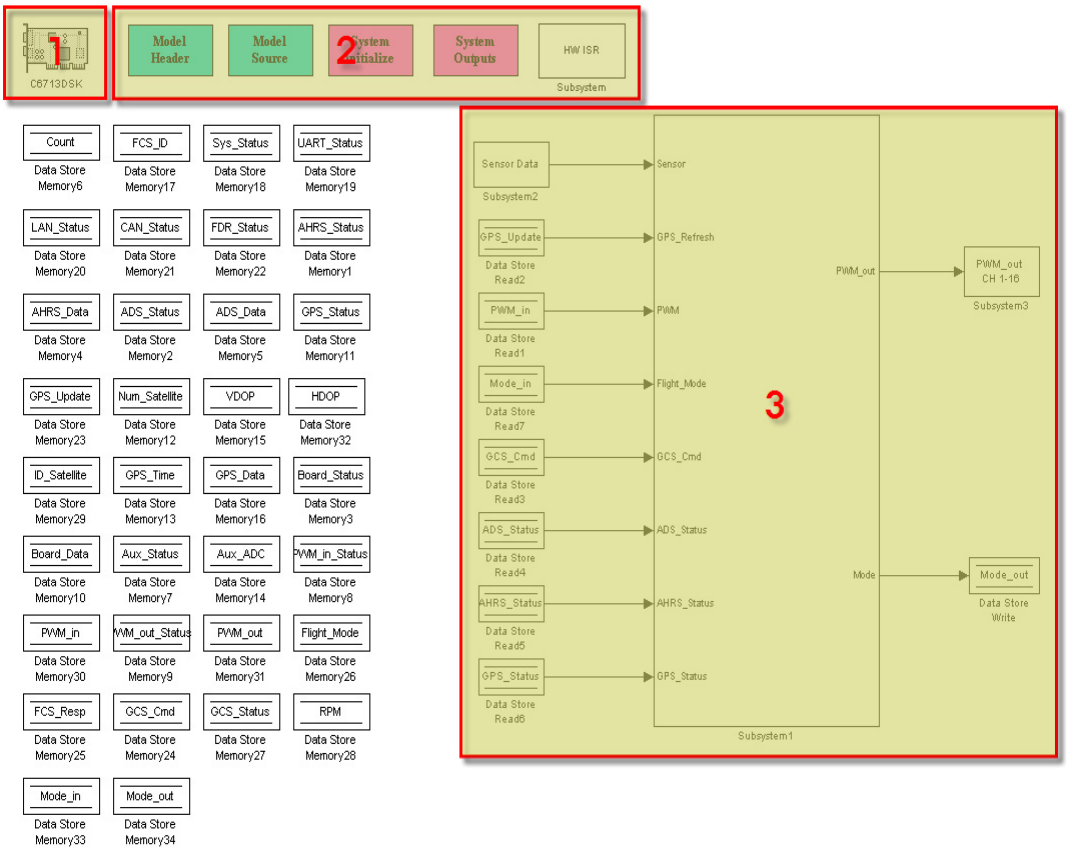

그림 9. Simulink Block으로 구성된 제어연산 시스템 프로그램

과 데이터 교환을 위한 DPRAM 드라이버가 필요하다. 이는 Simulink를 Simulation 하는 경우에는 동작하지 않으며, 하드웨어에 탑재할 경우에만 생성된 코드에 삽입되어 동작하는 부분이다. 그림 9에서 볼 수 있듯이 Module 2는 DPRAM 운용 드라이버 관련 초기화와 데이터 입·출력 그리고 Interrupt Service Routine 등으로 구성되어 있다.

2.4.3 Module 3

제어 알고리즘 부분으로 장치관리시스템에서 센서 인터페이스를 통해 모아진 데이터를 바탕으로 항공기의 자동항법 알고리즘과 임무 관련 프로그램으로 구성되어, 제어명령을 연산한다. 이 과정에서 얻어진 임무장비의 명령, 지상관제로 전달할 데이터, 작동기 명령 등을 다시 DPRAM을 통해 장치관리시스템에 전달한다.

제어연산시스템 프로그램의 코드검증 방안은 사전에 발표된 논문[10]에 상세히 기술되어 있다.

III. 결 론

본 논문에서는 소형 무인기의 통합형 비행제어시스템에 실시간 운영체제와 자동코드 생성 기술을 적용하는 방안에 대해 제시하였다. 실시간 운영체제는 다양한 센서시스템의 효율적인 관리와 임무장비를 위한 주변장치의 안정적인 운영에 주안점을 두고 개발하였다. 적용된 실시간 운영체제는 기본적인 센서 관리와 외부장치 연동을 위한 통신에 대해 최대 부하상태를 적용하여 실시간성과 기능요구를 검증하였다. 이렇게 개발된 탑재 프로그램은 통합 시스템의 능력을 최대한 활용할 수 있게 제어연산시스템을 지원한다.

개발된 통합 시스템의 제어연산시스템 프로그램은 하드웨어 개발자의 도움 없이도 제어 알고리즘 개발자 단독으로 제어알고리즘을 탑재할 수 있도록 개발하였다. 개발자에게 제공되는 Simulink 파일을 이용하면 모델기반 Auto Code Generation 기법을 이용해, 제어 알고리즘 개발자가 직접 컴파일러(Code Composer Studio)를 사용하지 않아

도 단독으로 통합 비행제어 시스템에 프로그램을 탑재하는 것이 가능하다. 결국 제어알고리즘 개발자는 익숙한 Matlab Simulink 환경에서 센서 및 임무장비 통합형 비행제어 시스템을 직접 활용할 수 있다.

후 기

이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(2010-0027511).

참고문헌

1) 윤승호, 김유단, "지상 목표물 추적을 위한 다수 무인항공기의 협력제어", 한국항공우주학회지, 제39권 제2호 2011.2, pp. 114~120

2) 김성환, 유창경, 박춘배, “거리 정보를 이용한 되먹임 선형화 기법 무인기 편대 비행제어”, 제어·로봇·시스템학회 논문지 , 15권 1호 2009.1, pp. 23~30

3) 유동일, 심현철, "편대 유도 법칙 및 초소형 비행체의 자동 편대 비행 구현", 한국항공우주학회지, 제39권 제2호 2011.2, pp. 121~127

4) 김성수, 김성환, 조상욱, 김창환, 정명진, 유창경, 최기영, 박춘배, “무인항공기용 고성능 비행제어시스템 개발”, 2008 제어자동화시스템 심포지엄, 2008.11.

5) 이기영, 김형근, 박주원, 홍천한, 김부민, 김병수, “소형 무인기용 센서통합형 비행제어 컴퓨터 개발,” 한국항공우주학회 추계학술대회, 2006.11.

6) 이성호, 박준현, 장성호, “무인기의 소형화 및 경량화를 위한 통합형 디지털비행조종컴퓨터 개발 ”, 한국항공우주학회 추계학술대회, 2010.11.

7) 하철근, 임재형, “출력기반 적응제어기법을 이용한 틸트로터 항공기의 회전익 모드 설계연구”, 한국항공우주학회지, 제38권 제3호 2010. 3, pp. 228~235

8) 김낙완, 유창선, 강영신, “피드백 선형화를 이용한 $\mathcal{L}_1$ 적응제어기법 연구”, 한국항공우주학회지, 제36권 제6호 2008.6, pp. 558~564

9) 문정호, 신성식, 최승기, 조신제, 노은정, “모델기반 개발기술을 적용한 무인항공기 비행제어 소프트웨어 개발”, 한국항공우주학회지, 제38권 제12호 2010.12, pp. 1217~1222

10) 조상욱, 최기영, “자동 코드생성을 이용한 무인기용 OFP의 검증에 관한 연구”, 한국항공우주학회지, 제37권 제4호 2009.4, pp. 359~366

11) 이태희, 조상, “실시간 운영체제를 탑재한 원격 제어 로봇 시스템”, 제어·로봇·시스템학회 논문지 , 10권 8호 2004.8, pp. 689~695

12) 안희준, 이동수, “Windows NT상에에서의 OPRoS 컴포넌트 스케줄러의 실시간성 분석 및 개선” 제어·로봇·시스템학회 논문지 , 17권 1호 2011.1, pp. 38~46

13) 민병문, 김성필, 김봉주, 김응태, 탁민제, “재형상 비행제어 시스템의 비행시험 결과 분석” 제어·로봇·시스템학회 논문지 , 14권 12호 2008. 12, pp. 1244~1252