

ARM Cortex-M 시리즈 2

ARM Cortex-M7

STM32F767 정보

(OK-STIM767 키트)

공학박사 윤덕용 제



- ARM Cortex-M7에 대한 기본적인 이해
- STM32F767VGT6 칩의 철저한 이해
- 암울 내장 I/O의 구조와 기능에 대한 이해
- 외부 I/O 인터페이스 설계
- STM32F767VGT6의 기본적인 프로그래밍
- UART에 대한 기본적인 프로그래밍 기술
- UART에 한글 디스플레이 기술
- 각종 I/O에 대한 제어 프로그래밍 기술
- MP3 플레이어 응용 프로그래밍 기술
- 실시간 디지털 신호처리 프로그래밍 기술
- OK-STM767 키트에서 프로그래밍 실습
- C언어를 사용한 프로그래밍 기술
(IAR EWARM C 컴파일러 사용)
- 예뮬레이터에 의한 프로그램 다운로드
(ST-LINK/V2 예뮬레이터를 사용)

OK-STM767 키트 받기

Ohm사

<http://www.ohm.co.kr>
오스본 홀딩스(주) 서울특별시 강남구 테헤란로 152

- STM32F767 정복

- https://book.naver.com/bookdb/book_detail.nhn?bid=11647852

- 이해도 85%

목차

제1장 STM32F767VGT6의 구조와 기능

1.1 ARM 마이크로프로세서의 개요11

1. ARM 마이크로프로세서의 역사 및 특징11

[휴게실] 마이크로프로세서와 마이크로컨트롤러21

[휴게실] CPU에서 레지스터 중심구조와 하버드 구조22

2. Cortex-M3 마이크로컨트롤러의 특징과 종류23

3. Cortex-M4 마이크로컨트롤러의 특징과 종류34

4. Cortex-M7 마이크로컨트롤러의 특징과 종류47

5. Cortex-M0 마이크로컨트롤러의 특징과 종류50

6. Cortex-M0+ 마이크로컨트롤러의 특징과 종류58

1.2 STM32F767VGT6의 기본 구조와 기능66

1. STM32F767VGT6의 특징66

2. STM32F767VGT6의 외부 구조68

3. STM32F767VGT6의 내부 구조71

[휴게실] EEPROM과 플래시 메모리77

4. STM32F767VGT6의 메모리 구조78

1.3 STM32F767VGT6의 기본 시스템 제어기92

1. 전력관리 제어기(PWR)92

2. 리셋 및 클럭 제어기(RCC) 103

3. 시스템 설정 제어기(SYSCFG) 130

4. 시스템 제어 블록(SCB) 134

5. 시스템 타이머(SysTick) 139

6. 인터럽트 제어기(NVIC, EXTI) 142

7. 디버그 지원 장치(DBG) 153

[휴게실] I/O 제어 레지스터의 비트 속성156

2.1 병렬 I/O 포트(GPIO) 159

1. GPIO의 개요 159

2. GPIO의 구조와 동작 166

3. GPIO 관련 I/O 제어 레지스터 169

2.2 A/D 컨버터(ADC) 174

1. A/D 컨버터의 개요 174

2. A/D 컨버터의 구조와 동작 176

3. A/D 컨버터 관련 I/O 제어 레지스터 192

2.3 D/A 컨버터(DAC) 202

1. D/A 컨버터의 개요 202

2. D/A 컨버터의 구조와 동작 203

3. D/A 컨버터 관련 I/O 제어 레지스터 208

2.4 워치독 타이머(IWDG, WWDG) 214

1. 독립 워치독 타이머(IWDG)의 구조와 동작 214

2. IWDG 관련 I/O 제어 레지스터 217

3. 윈도우 워치독 타이머(WWDG)의 구조와 동작 219

4. WWDG 관련 I/O 제어 레지스터 222

2.5 고성능 제어 타이머(TIM1/8) 224

1. STM32F767VGT6 타이머의 개요 224

2. 타이머 TIM1/8의 개요 225

3. 타이머 TIM1/8의 구조와 동작 225

4. 타이머 TIM1/8 관련 I/O 제어 레지스터 255

2.6 범용 타이머(TIM2/3/4/5) 278

1. 타이머 TIM2/3/4/5의 개요 278

2. 타이머 TIM2/3/4/5의 구조와 동작 278

3. 타이머 TIM2/3/4/5 관련 I/O 제어 레지스터 280

2.7범용 타이머(TIM9/12) 298

1. 타이머 TIM9/12의 개요 298
2. 타이머 TIM9/12의 구조와 동작 298
3. 타이머 TIM9/12 관련 I/O 제어 레지스터 300

2.8범용 타이머(TIM10/11/13/14) 308

1. 타이머 TIM10/11/13/14의 개요 308
2. 타이머 TIM10/11/13/14의 구조와 동작 308
3. 타이머 TIM10/11/13/14 관련 I/O 제어 레지스터 310

2.9기본 타이머(TIM6/7) 315

1. 타이머 TIM6/7의 개요 315
2. 타이머 TIM6/7의 구조와 동작 315
3. 타이머 TIM6/7 관련 I/O 제어 레지스터 319

2.10동기 및 비동기 직렬통신 포트(USART) 323

1. USART 직렬통신 포트의 개요 323
2. USART 직렬통신 포트의 구조와 동작 325
3. USART 직렬통신 포트 관련 I/O 제어 레지스터 339

4. RS-232C 직렬통신 346

[휴게실] ASCII 코드 352

2.11동기식 직렬통신 포트(SPI) 353

1. SPI 직렬통신의 개요 353
2. SPI 직렬통신 포트의 구조와 동작 356
3. SPI 직렬통신 포트 관련 I/O 제어 레지스터 367

2.12동기식 직렬통신 포트(I2C) 372

1. I2C 직렬통신의 개요 372
2. I2C 직렬통신 포트의 구조와 동작 377
3. I2C 직렬통신 포트 관련 I/O 제어 레지스터 394

제3장 OK-STM767 키트 및 개발 툴

3.1OK-STM767 키트의 구조와 기능 403

1. OK-STM767 키트의 개요 및 사양 403
2. OK-STM767 키트의 하드웨어 구조 405
3. TFT-32A 보드의 하드웨어 구조 416
4. OK-STM767 키트의 조립 및 테스트 422
5. RS-232C 통신 케이블의 제작 427

3.2Cortex-M용 에뮬레이터 ST-LINK/V2 429

1. 하드웨어 및 소프트웨어 개발 툴 429
2. ST-LINK/V2 에뮬레이터 434

3.3IAR EWARM 컴파일러의 설치 및 사용 443

1. IAR EWARM 프로그램의 설치 443
2. 예제 프로그램의 설치 449
3. IAR EWARM의 환경 설정 449
- [휴게실] 인텔 HEX 파일의 형식 462
4. IAR EWARM의 주요 기능 요약 463
5. 주요 내장함수 및 헤더파일 479

제4장 C언어 프로그래밍

4.1IAR EWARM을 사용한 C언어 프로그래밍 기초	523
4.2기본적인 C언어 프로그래밍 기법	555
4.3텍스트형 LCD 모듈 응용 프로그램	575
[휴게실] 텍스트 LCD 모듈과 그래픽형 LCD 모듈의 차이점	592
4.4키입력 및 인터럽트 처리 프로그램	608
4.5TFT-LCD 모듈 영문 ASCII 출력 프로그램	623
4.6TFT-LCD 모듈 한글 출력 프로그램	671
4.7TFT-LCD 모듈 그래픽 출력 프로그램	729
4.8터치스크린 입력 프로그램	743
4.9SysTick 타이머 인터럽트 응용 프로그램	769
4.10TV 리모컨 응용 프로그램	785
4.11타이머 응용 프로그램	811
4.12타이머를 이용한 PWM 제어 프로그램	819
4.13A/D 컨버터 응용 프로그램	826
4.14D/A 컨버터 응용 프로그램	851
4.15RS-232C 직렬통신 프로그램	880
4.16SPI 및 I2C 직렬통신 프로그램	890
4.17DS3234를 이용한 시계 프로그램	904
4.18디지털 신호처리(FFT) 프로그램	959
4.19SD 카드를 이용한 전자앨범 프로그램	985
4.20MP3 플레이어 프로그램	1023
참고문헌 및 저자소개	1064

- 윤덕용교수님 책은 워낙 유명해서 익히 알고 있었고, Firmware의 레지스터 제어 방식을 제대로 공부하기 위해서 보았다. 책의 앞부분은 STM32F767 MCU의 각각의 아키텍처와 그 안의 레지스터에 대해 교과서식 + 저자의 풀어서 쓴 설명 반반으로 전개된다. 뒷부분은 각각의 Peripheral 에 대한 보드 기반의 예제를 담고 있다. 나는 이 책에서 사용하는 보드도 직접 ohm사에서 구매해서 공부했고, 레지스터를 공부하는 데 어느 정도 성공했다고 생각한다.

그러나 최근의 Firmware 개발 트렌드는 레지스터를 직접 건드리기 보다는 ST 사의 MCU의 경우 CubeMX의 GUI에 의해 레지스터 셋팅을 굉장히 편리하게 하고, HAL라이브러리를 사용해서 응용 소프트웨어를 편하게 개발하고 있는 추세이다. 레지스터를 직접 제어해야 하느냐, 아니면 HAL라이브러리를 사용해야 하느냐에 대해서는 인터넷 상의 논쟁을 몇 번 보았지만 난 HAL 라이브러리를 사용하는 것이 맞다고 생각한다. 컴퓨터 아키텍처를 직접 설계하는 사람이라면 레지스터를 잘 이해하고 있기 때문에 레지스터 제어를 통해 firmware를 하는 것이 맞지만, 우리 같은 일반 엔지니어는 컴퓨터 아키텍처를 설계해 본 일도 없을 뿐더러 개발을 빠르게 하기 위해서는 이미 검증된 HAL 라이브러리를 사용하는 것이 훨씬 유리하기 때문이다.

만약 레지스터를 직접 제어한다면 목적에 맞게 32bit나 되는 레지스터 각각을 셋팅해야 하고, 함수까지 자기가 직접 만들면서 개발을 해야한다. MCU를 직접 만드는 사람이 아니고서야 자기가 만든 함수보다는 당연히 ST사에서 제공하는 HAL 라이브러리가 신뢰성, 완성도가 있을 수 밖에 없다.

그러므로 나는 이제 Firmware 개발은 코드 자동 생성툴을 사용하고, HAL 라이브러리를 사용해서 개발을 하는 것이 맞다고 생각한다. 그러나 주의할 점은 그것에만 의존해서는 안된다는 것이다. 라이브러리를 사용하다가 문제가 터졌을 때 라이브러리 내부를 분석하며 수정할 수도 있어야 하고, 라이브러리를 제대로 이해하는 것은 해당 MCU 아키텍처에 대한 공부가 되므로 Firmware 개발을 더 잘할 수 있는 기반이 된다고 생각한다.