AVR-GCC Compiler in WinAVR

Jee-Hwan Ryu

School of Mechanical Engineering
Korea University of Technology and Education

AVR-GCC의 개요

- AVR용의 GCC 컴파일러
- GCC는 C언어 및 C++언어를 지원하는 ANSI C 컴파일러
- 공개용이지만 매우 우수한 성능과 안정성을 가지고 있음
- 전세계 많은 지원자들에 의하여 지속적으로 발전
- Windows 운영체제 IBM PC (host platform), AVR 마이크로콘트롤러 (Target platform) -> cross compiler
- http://winavr.sourceforge.net/에서 최신버젼 다운로드
- WinAVR
  - Programmers Notepad
  - MFile

SRAM 에서의 변수 및 상수

- AVR-GCC에서 특별히 속성을 지정하지 않고 변수를 정의하면 MCU내부의 SRAM에 저장된다.
- 외부 메모리에 저장되는 경우에 비하여 빠르게 액세스 가능
- <inttypes.h>에 정의, <stdio.h>에서 자동으로 인클루드 시킴

일반적인 데이터형 표현	AVR-GCC 데이터형 표현	바이트 수	표현 범위
signed char	int8_t	1	-128~127
unsigned char	uint8_t	1	0~255
signed int	int16_t	2	-32768 ~ 32767
unsigned int	uint16_t	2	0 ~ 65535
singed long	int32_t	4	-2147483648 ~ 2147483647
unsigned long	uint32_t	4	0 ~ 4294967295
signed long long	int64_t	8	-9.22x10^18 ~ 9.22x10^18
unsigned long long	uint64_t	8	0 ~ 1.844x10^19
	intptr_t	2	int16_t와 같음
	uintptr_t	2	uint16_t와 같음

변수 및 상수 정의

- uint8_t val=10; //과 같이 정의 후
  val = 2; // 변경 가능
- const uint8_t conval = 10; //상수로 정의한 것은 읽기만 가능
- register uint8_t regval; //변수를 레지스터에 할당하면 액세스 속도가 더욱 빠르므로 자주 사용하는 변수에 유용

- AVR-GCC 전용의 데이터 표현 방법보다는 가급적 일반적인 데이터 표현 방법을 사용하는 것을 권장

I/O 레지스터 및 병렬 I/O 포트의 액세스

- unsigned char inp(unsinged char port);
  outp(unsigned char val, unsinged char port); //지정된 port I/O레지스터로부터 1바이트의 데이터를 읽거나 출력한다.
- void sbi(unsinged char port, unsigned char bit);
  void cbi(unsinged char port, unsigned char bit);
  //지정된 I/O 레지스터에서 특정 비트를 1로 세트 시키거나 0으로 클리어 시킨다.
- unsigned char bit_is_set(unsinged char port, unsinged char bit);
  unsinged char bit_is_clear(unsinged char port, unsinged char bit); //지정된 I/O레지스터의 특정 비트가 1이나 0으로 설정되어 있는지 검사하여 1(true) 혹은 0(false)출력

I/O 레지스터 및 병렬 I/O 포트의 액세스

- I/O레지스터 중 병렬 I/O레지스터 액세스용 레지스터
  - DDRn: 입/출력 방향 설정, 0xFF(모두출력), 0x00(모두입력)
  - PORTn: 출력으로 정의된 포트에 데이터 출력시
  - PINn: 입력으로 정의된 포트에 데이터 입력시
- outp(0x80,PORTB);
- unsigned char aaa;
  aaa=inp(PINF);
- unsigned char aaa;
  DDRB=0xFF;
  DDRF=0x00;
  aaa=PINF;
  PORTB=0x80;

인터럽트 프로그래밍

- #include <signal.h>
  SIGNAL(sig_name)
  {
  //interrupt service routine
  }
- #include <signal.h>
  INTERRUPT(sig_name)
  {
  //interrupt service routine
  }
- sig_name <표 3.7.2> 참조
- SIGNAL은 이 함수 수행 동안 다른 인터럽트 허용 안 함, INTERRUPT는 다른 인터럽트 허용
- <interrupt.h> 내에 인터럽트 허용 금지하는 함수
  - sei();
  - cli();

인라인 어셈블 프로그래밍

- C언어 소스에서 어셈블리 명령을 사용하고 싶을 경우
  asm volatile("assembly instruction ₩n₩t"
  "assembly instruction ₩n₩t"
  .....
  "assembly instruction ₩n₩t");
- 인라인 어셈블 루틴에서 C소스의 변수명을 사용하려면 반드시 글로벌 변수여야 한다.

실험보고서 작성법

- 실험 목적
- 이론적 배경
- 실험과정
- 실험결과
- 고찰