KEYWORDS:
C, BITFIELD, I/O, IAR ICCA90

Efficient I/O Handling with Bitfields

Introduction

Bitfields provide a simple user interface, but can in many cases cause the compiler to generate inefficient code. This design note describes how to access each single bit in the AVR I/O registers with bitfields without any code overhead in the IAR icca90 family of compilers. Bitfields allow easy change of functionality of the I/O ports.

Overview

This example code shows how to configure and use bitfields for I/O ports

```
// First define the bits in the port
typedef struct _bit_struct
{
  unsigned char bit0 : 1;
  unsigned char bit1 : 1;
  unsigned char bit2 : 1;
  unsigned char bit3 : 1;
  unsigned char bit4 : 1;
  unsigned char bit5 : 1;
  unsigned char bit6 : 1;
  unsigned char bit7 : 1;
} bit_field;

// Define macro to get the value of each bit
#define GET_BITFIELD(addr) (*((volatile bit_field*) (addr)))

// Define port addresses (Included in the ioxxxx.h file)
#define PORTB  0x38
#define PINB   0x36

// Define functions for each bit of the I/O ports
#define LED1      GET_BITFIELD(PORTB).bit0  // Outputs use the PORT address
#define LED2      GET_BITFIELD(PORTB).bit1
#define BUTTON    GET_BITFIELD(PINB).bit4   // Inputs use the PIN address

// User code
void C_task main(void)
{
 if(0 == BUTTON)                  // Test for a single input bit = 0
```

```
 {
     LED1 = 1;                    // Set the value of one output
     LED2 = 0;                    // Clear the value of another output
 }
 else if(1 == BUTTON)             // Test for a single input bit = 1
 {
     LED1 = 0;                    // Clear the value of a single bit
 }
}
```