```
/*****************************************************************************
*
* Atmel Corporation
*
* File              : TWI_Slave.h
* Compiler          : IAR EWAAVR 2.28a/3.10c
* Revision          : $Revision: 2475 $
* Date              : $Date: 2007-09-20 12:00:43 +0200 (to, 20 sep 2007) $
* Updated by        : $Author: mlarsson $
*
* Support mail      : avr@atmel.com
*
* Supported devices : All devices with a TWI module can be used.
*                     The example is written for the ATmega16
*
* AppNote           : AVR311 - TWI Slave Implementation
*
* Description       : Header file for TWI_slave.c
*                     Include this file in the application.
*
*****************************************************************************/
/*! \page MISRA
 *
 * General disabling of MISRA rules:
 * * (MISRA C rule 1) compiler is configured to allow extensions
 * * (MISRA C rule 111) bit fields shall only be defined to be of type unsigned int
or signed int
 * * (MISRA C rule 37) bitwise operations shall not be performed on signed integer
types
 * As it does not work well with 8bit architecture and/or IAR

 * Other disabled MISRA rules
 * * (MISRA C rule 109) use of union - overlapping storage shall not be used
 * * (MISRA C rule 61) every non-empty case clause in a switch statement shall be
terminated with a break statement
*/

/*****************************************************************************
  TWI Status/Control register definitions
*****************************************************************************/

#define TWI_BUFFER_SIZE 4      // Reserves memory for the drivers transceiver
buffer.
                               // Set this to the largest message size that will be
sent including address byte.

/*****************************************************************************
  Global definitions
```

```
*************************************************************************/

union TWI_statusReg_t                       // Status byte holding flags.
{
    unsigned char all;
    struct
    {
        unsigned char lastTransOK:1;
        unsigned char RxDataInBuf:1;
        unsigned char genAddressCall:1;                         // TRUE = General
call, FALSE = TWI Address;
        unsigned char unusedBits:5;
    };
};

extern union TWI_statusReg_t TWI_statusReg;


/*************************************************************************
  Function definitions
*************************************************************************/
void TWI_Slave_Initialise( unsigned char );
unsigned char TWI_Transceiver_Busy( void );
unsigned char TWI_Get_State_Info( void );
void TWI_Start_Transceiver_With_Data( unsigned char * , unsigned char );
void TWI_Start_Transceiver( void );
unsigned char TWI_Get_Data_From_Transceiver( unsigned char *, unsigned char );

/*************************************************************************
  Bit and byte definitions
*************************************************************************/
#define TWI_READ_BIT  0   // Bit position for R/W bit in "address byte".
#define TWI_ADR_BITS  1   // Bit position for LSB of the slave address bits in the
init byte.
#define TWI_GEN_BIT   0   // Bit position for LSB of the general call bit in the
init byte.

#define TRUE          1
#define FALSE         0

/*************************************************************************
  TWI State codes
*************************************************************************/
// General TWI Master staus codes
#define TWI_START                  0x08  // START has been transmitted
#define TWI_REP_START              0x10  // Repeated START has been transmitted
#define TWI_ARB_LOST               0x38  // Arbitration lost
```

```
// TWI Master Transmitter staus codes
#define TWI_MTX_ADR_ACK            0x18  // SLA+W has been tramsmitted and ACK
received
#define TWI_MTX_ADR_NACK           0x20  // SLA+W has been tramsmitted and NACK
received
#define TWI_MTX_DATA_ACK           0x28  // Data byte has been tramsmitted and ACK
received
#define TWI_MTX_DATA_NACK          0x30  // Data byte has been tramsmitted and NACK
received

// TWI Master Receiver staus codes
#define TWI_MRX_ADR_ACK            0x40  // SLA+R has been tramsmitted and ACK
received
#define TWI_MRX_ADR_NACK           0x48  // SLA+R has been tramsmitted and NACK
received
#define TWI_MRX_DATA_ACK           0x50  // Data byte has been received and ACK
tramsmitted
#define TWI_MRX_DATA_NACK          0x58  // Data byte has been received and NACK
tramsmitted

// TWI Slave Transmitter staus codes
#define TWI_STX_ADR_ACK            0xA8  // Own SLA+R has been received; ACK has
been returned
#define TWI_STX_ADR_ACK_M_ARB_LOST 0xB0  // Arbitration lost in SLA+R/W as Master;
own SLA+R has been received; ACK has been returned
#define TWI_STX_DATA_ACK           0xB8  // Data byte in TWDR has been transmitted;
ACK has been received
#define TWI_STX_DATA_NACK          0xC0  // Data byte in TWDR has been transmitted;
NOT ACK has been received
#define TWI_STX_DATA_ACK_LAST_BYTE 0xC8  // Last data byte in TWDR has been
transmitted (TWEA = “0”); ACK has been received

// TWI Slave Receiver staus codes
#define TWI_SRX_ADR_ACK            0x60  // Own SLA+W has been received ACK has been
returned
#define TWI_SRX_ADR_ACK_M_ARB_LOST 0x68  // Arbitration lost in SLA+R/W as Master;
own SLA+W has been received; ACK has been returned
#define TWI_SRX_GEN_ACK            0x70  // General call address has been received;
ACK has been returned
#define TWI_SRX_GEN_ACK_M_ARB_LOST 0x78  // Arbitration lost in SLA+R/W as Master;
General call address has been received; ACK has been returned
#define TWI_SRX_ADR_DATA_ACK       0x80  // Previously addressed with own SLA+W;
data has been received; ACK has been returned
#define TWI_SRX_ADR_DATA_NACK      0x88  // Previously addressed with own SLA+W;
data has been received; NOT ACK has been returned
#define TWI_SRX_GEN_DATA_ACK       0x90  // Previously addressed with general call;
data has been received; ACK has been returned
#define TWI_SRX_GEN_DATA_NACK      0x98  // Previously addressed with general call;
```

```
data has been received; NOT ACK has been returned
#define TWI_SRX_STOP_RESTART       0xA0  // A STOP condition or repeated START
condition has been received while still addressed as Slave

// TWI Miscellaneous status codes
#define TWI_NO_STATE               0xF8  // No relevant state information available;
TWINT = “0”
#define TWI_BUS_ERROR              0x00  // Bus error due to an illegal START or
STOP condition
```