```
[code]#ifndef SOFTI2CMASTER_H_

#define SOFTI2DMASTER_H_

#include <avr/io.h>
#include <Arduino.h>

#if defined(ARDUINO_AVR_FLORA8) || defined(ARDUINO_AVR_LILYPAD_USB)
#define SDA_PORT PORTD
#define SDA_PIN 1
#define SCL_PORT PORTD
#define SCL_PIN 0
#endif

#ifdef ARDUINO_AVR_UNO
#define SDA_PORT PORTC
#define SDA_PIN 4
#define SCL_PORT PORTC
#define SCL_PIN 5
#endif

#define I2C_TIMEOUT 100
#define I2C_NOINTERRUPT 0
#define I2C_SLOWMODE 1
#define FAC 1
#define I2C_CPUFREQ (F_CPU/FAC)

#ifndef _SOFTI2C_H
#define _SOFTI2C_H   1


boolean __attribute__ ((noinline)) i2c_init(void);
```

```
// Start transfer function: <addr> is the 8-bit I2C address (including the R/W
// bit).
// Return: true if the slave replies with an "acknowledge", false otherwise
bool __attribute__ ((noinline)) i2c_start(uint8_t addr);


void  __attribute__ ((noinline)) i2c_start_wait(uint8_t addr);


// Return: true if the slave replies with an "acknowledge", false otherwise
bool __attribute__ ((noinline)) i2c_rep_start(uint8_t addr);

// Issue a stop condition, freeing the bus.
void __attribute__ ((noinline)) i2c_stop(void) asm("ass_i2c_stop");


bool __attribute__ ((noinline)) i2c_write(uint8_t value) asm("ass_i2c_write");


uint8_t __attribute__ ((noinline)) i2c_read(bool last);


#ifndef I2C_CPUFREQ
#define I2C_CPUFREQ F_CPU
#endif


#ifndef I2C_FASTMODE
#define I2C_FASTMODE 0
#endif


#ifndef I2C_SLOWMODE
#define I2C_SLOWMODE 0
```

```
#endif

// if I2C_NOINTERRUPT is 1,
#ifndef I2C_NOINTERRUPT
#define I2C_NOINTERRUPT 0
#endif


#ifndef I2C_TIMEOUT
#define I2C_TIMEOUT 0
#else
#if I2C_TIMEOUT > 10000
#error I2C_TIMEOUT is too large
#endif
#endif

#define I2C_TIMEOUT_DELAY_LOOPS (I2C_CPUFREQ/1000UL)*I2C_TIMEOUT/4000UL
#if I2C_TIMEOUT_DELAY_LOOPS < 1
#define I2C_MAX_STRETCH 1
#else
#if I2C_TIMEOUT_DELAY_LOOPS > 60000UL
#define I2C_MAX_STRETCH 60000UL
#else
#define I2C_MAX_STRETCH I2C_TIMEOUT_DELAY_LOOPS
#endif
#endif

#if I2C_FASTMODE
#define I2C_DELAY_COUNTER (((I2C_CPUFREQ/400000L)/2-19)/3)
#else
#if I2C_SLOWMODE
#define I2C_DELAY_COUNTER (((I2C_CPUFREQ/25000L)/2-19)/3)
#else
#define I2C_DELAY_COUNTER (((I2C_CPUFREQ/100000L)/2-19)/3)
```

```
#endif
#endif

// Table of I2C bus speed in kbit/sec:
// CPU clock:          1MHz   2MHz    4MHz   8MHz   16MHz   20MHz
// Fast I2C mode         40     80     150    300     400     400
// Standard I2C mode     40     80     100    100     100     100
// Slow I2C mode         25     25      25     25      25      25

// constants for reading & writing
#define I2C_READ    1
#define I2C_WRITE   0

// map the IO register back into the IO address space
#define SDA_DDR       (_SFR_IO_ADDR(SDA_PORT) - 1)
#define SCL_DDR       (_SFR_IO_ADDR(SCL_PORT) - 1)
#define SDA_OUT       _SFR_IO_ADDR(SDA_PORT)
#define SCL_OUT       _SFR_IO_ADDR(SCL_PORT)
#define SDA_IN (_SFR_IO_ADDR(SDA_PORT) - 2)
#define SCL_IN (_SFR_IO_ADDR(SCL_PORT) - 2)

#ifndef __tmp_reg__
#define __tmp_reg__ 0
#endif


// Internal delay functions.
void __attribute__ ((noinline)) i2c_delay_half(void) asm("ass_i2c_delay_half");
void __attribute__ ((noinline)) i2c_wait_scl_high(void) asm("ass_i2c_wait_scl_high");

void  i2c_delay_half(void)
{ // function call 3 cycles => 3C
#if I2C_DELAY_COUNTER < 1
  __asm__ __volatile__ (" ret");
```

```
  // 7 cycles for call and return
#else
  __asm__ __volatile__
    (
     " ldi      r25, %[DELAY]          ;load delay constant   ;; 4C \n\t"
     "_Lidelay: \n\t"
     " dec r25                        ;decrement counter     ;; 4C+xC \n\t"
     " brne _Lidelay                                  ;;5C+(x-1)2C+xC\n\t"
     " ret                                          ;; 9C+(x-1)2C+xC = 7C+xC"
     : : [DELAY] "M" I2C_DELAY_COUNTER : "r25");
  // 7 cycles + 3 times x cycles
#endif
}

void i2c_wait_scl_high(void)
{
#if I2C_TIMEOUT <= 0
  __asm__ __volatile__
    ("_Li2c_wait_stretch: \n\t"
     " sbis %[SCLIN],%[SCLPIN] ;wait for SCL high \n\t"
     " rjmp _Li2c_wait_stretch \n\t"
     " cln                          ;signal: no timeout \n\t"
     " ret "
     : : [SCLIN] "I" (SCL_IN), [SCLPIN] "I" (SCL_PIN));
#else
  __asm__ __volatile__
    ( " ldi     r27, %[HISTRETCH]       ;load delay counter \n\t"
      " ldi     r26, %[LOSTRETCH] \n\t"
      "_Lwait_stretch: \n\t"
      " clr     __tmp_reg__             ;do next loop 255 times \n\t"
      "_Lwait_stretch_inner_loop: \n\t"
      " rcall   _Lcheck_scl_level       ;call check function   ;; 12C \n\t"
      " brpl    _Lstretch_done          ;done if N=0           ;; +1 = 13C\n\t"
      " dec     __tmp_reg__             ;dec inner loop counter;; +1 = 14C\n\t"
```

```
    " brne    _Lwait_stretch_inner_loop                    ;; +2 = 16C\n\t"
    " sbiw    r26,1               ;dec outer loop counter \n\t"
    " brne    _Lwait_stretch         ;continue with outer loop \n\t"
    " sen                         ;timeout -> set N-bit=1 \n\t"
    " rjmp _Lwait_return            ;and return with N=1\n\t"
    "_Lstretch_done:               ;SCL=1 sensed \n\t"
    " cln                         ;OK -> clear N-bit \n\t"
    " rjmp _Lwait_return            ; and return with N=0 \n\t"

    "_Lcheck_scl_level:                                ;; call = 3C\n\t"
    " cln                                          ;; +1C = 4C \n\t"
    " sbic %[SCLIN],%[SCLPIN]      ;skip if SCL still low ;; +2C = 6C \n\t"
    " rjmp    _Lscl_high                               ;; +0C = 6C \n\t"
    " sen                                           ;; +1 = 7C\n\t "
    "_Lscl_high: "
    " nop                                           ;; +1C = 8C \n\t"
    " ret                         ;return N-Bit=1 if low ;; +4 = 12C\n\t"

    "_Lwait_return:"
    : : [SCLIN] "I" (SCL_IN), [SCLPIN] "I" (SCL_PIN),
 [HISTRETCH] "M" (I2C_MAX_STRETCH>>8),
 [LOSTRETCH] "M" (I2C_MAX_STRETCH&0xFF)
    : "r26", "r27");
#endif
}



boolean i2c_init(void)
{
  __asm__ __volatile__
   (" cbi     %[SDADDR],%[SDAPIN]    ;release SDA \n\t"
    " cbi     %[SCLDDR],%[SCLPIN]    ;release SCL \n\t"
    " cbi     %[SDAOUT],%[SDAPIN]    ;clear SDA output value \n\t"
    " cbi     %[SCLOUT],%[SCLPIN]    ;clear SCL output value \n\t"
```

```
    " clr     r24                 ;set return value to false \n\t"
    " clr     r25                 ;set return value to false \n\t"
    " sbis    %[SDAIN],%[SDAPIN]     ;check for SDA high\n\t"
    " ret                         ;if low return with false \n\t"
    " sbis    %[SCLIN],%[SCLPIN]     ;check for SCL high \n\t"
    " ret                         ;if low return with false \n\t"
    " ldi     r24,1                ;set return value to true \n\t"
    " ret "
    : :
      [SCLDDR] "I"  (SCL_DDR), [SCLPIN] "I" (SCL_PIN),
      [SCLIN] "I" (SCL_IN), [SCLOUT] "I" (SCL_OUT),
      [SDADDR] "I"  (SDA_DDR), [SDAPIN] "I" (SDA_PIN),
      [SDAIN] "I" (SDA_IN), [SDAOUT] "I" (SDA_OUT));
  return true;
}

bool  i2c_start(uint8_t addr)
{
  __asm__ __volatile__
    (
#if I2C_NOINTERRUPT
    " cli                         ;clear IRQ bit \n\t"
#endif
    " sbis    %[SCLIN],%[SCLPIN]     ;check for clock stretching slave\n\t"
    " rcall   ass_i2c_wait_scl_high   ;wait until SCL=H\n\t"
    " sbi     %[SDADDR],%[SDAPIN]    ;force SDA low  \n\t"
    " rcall   ass_i2c_delay_half     ;wait T/2 \n\t"
    " rcall   ass_i2c_write          ;now write address \n\t"
    " ret"
    : : [SDADDR] "I"  (SDA_DDR), [SDAPIN] "I" (SDA_PIN),
      [SCLIN] "I" (SCL_IN),[SCLPIN] "I" (SCL_PIN));
  return true; // we never return here!
}
```

```
bool  i2c_rep_start(uint8_t addr)
{
  __asm__ __volatile__

    (
#if I2C_NOINTERRUPT
     " cli \n\t"
#endif
     " sbi %[SCLDDR],%[SCLPIN] ;force SCL low \n\t"
     " rcall ass_i2c_delay_half ;delay  T/2 \n\t"
     " cbi %[SDADDR],%[SDAPIN] ;release SDA \n\t"
     " rcall ass_i2c_delay_half ;delay T/2 \n\t"
     " cbi %[SCLDDR],%[SCLPIN] ;release SCL \n\t"
     " rcall ass_i2c_delay_half ;delay  T/2 \n\t"
     " sbis     %[SCLIN],%[SCLPIN]      ;check for clock stretching slave\n\t"
     " rcall    ass_i2c_wait_scl_high   ;wait until SCL=H\n\t"
     " sbi %[SDADDR],%[SDAPIN] ;force SDA low \n\t"
     " rcall ass_i2c_delay_half ;delay T/2 \n\t"
     " rcall    ass_i2c_write       \n\t"
     " ret"
     : : [SCLDDR] "I"  (SCL_DDR), [SCLPIN] "I" (SCL_PIN),[SCLIN] "I" (SCL_IN),
         [SDADDR] "I"  (SDA_DDR), [SDAPIN] "I" (SDA_PIN));
  return true; // just to fool the compiler
}

void  i2c_start_wait(uint8_t addr)
{
 __asm__ __volatile__
   (
    " push r24                  ;save original parameter \n\t"
    "_Li2c_start_wait1: \n\t"
    " pop       r24                 ;restore original parameter\n\t"
    " push      r24                 ;and save again \n\t"
#if I2C_NOINTERRUPT
```

```
   " cli                        ;disable interrupts \n\t"
#endif
   " sbis    %[SCLIN],%[SCLPIN]     ;check for clock stretching slave\n\t"
   " rcall    ass_i2c_wait_scl_high   ;wait until SCL=H\n\t"
   " sbi %[SDADDR],%[SDAPIN] ;force SDA low \n\t"
   " rcall ass_i2c_delay_half ;delay T/2 \n\t"
   " rcall ass_i2c_write       ;write address \n\t"
   " tst r24       ;if device not busy -> done \n\t"
   " brne _Li2c_start_wait_done \n\t"
   " rcall ass_i2c_stop       ;terminate write & enable IRQ \n\t"
   " rjmp _Li2c_start_wait1 ;device busy, poll ack again \n\t"
   "_Li2c_start_wait_done: \n\t"
   " pop      __tmp_reg__           ;pop off orig argument \n\t"
   " ret "
   : : [SDADDR] "I"  (SDA_DDR), [SDAPIN] "I" (SDA_PIN),
     [SCLIN] "I" (SCL_IN),[SCLPIN] "I" (SCL_PIN));
}

void  i2c_stop(void)
{
  __asm__ __volatile__
   (
    " sbi     %[SCLDDR],%[SCLPIN]     ;force SCL low \n\t"
    " sbi     %[SDADDR],%[SDAPIN]     ;force SDA low \n\t"
    " rcall    ass_i2c_delay_half     ;T/2 delay \n\t"
    " cbi     %[SCLDDR],%[SCLPIN]     ;release SCL \n\t"
    " rcall    ass_i2c_delay_half     ;T/2 delay \n\t"
    " sbis    %[SCLIN],%[SCLPIN]     ;check for clock stretching slave\n\t"
    " rcall    ass_i2c_wait_scl_high   ;wait until SCL=H\n\t"
    " cbi     %[SDADDR],%[SDAPIN]     ;release SDA \n\t"
    " rcall    ass_i2c_delay_half \n\t"
#if I2C_NOINTERRUPT
    " sei                        ;enable interrupts again!\n\t"
#endif
```

```
    : : [SCLDDR] "I"  (SCL_DDR), [SCLPIN] "I" (SCL_PIN), [SCLIN] "I" (SCL_IN),
       [SDADDR] "I"  (SDA_DDR), [SDAPIN] "I" (SDA_PIN));
}

bool i2c_write(uint8_t value)
{
  __asm__ __volatile__
    (
    " sec                        ;set carry flag \n\t"
    " rol     r24                ;shift in carry and shift out MSB \n\t"
    " rjmp _Li2c_write_first \n\t"
    "_Li2c_write_bit:\n\t"
    " lsl     r24                ;left shift into carry ;; 1C\n\t"
    "_Li2c_write_first:\n\t"
    " breq    _Li2c_get_ack      ;jump if TXreg is empty;; +1 = 2C \n\t"
    " sbi     %[SCLDDR],%[SCLPIN]    ;force SCL low       ;; +2 = 4C \n\t"
    " nop \n\t"
    " nop \n\t"
    " nop \n\t"
    " brcc    _Li2c_write_low                ;;+1/+2=5/6C\n\t"
    " nop                                    ;; +1 = 7C \n\t"
    " cbi %[SDADDR],%[SDAPIN]      ;release SDA       ;; +2 = 9C \n\t"
    " rjmp     _Li2c_write_high              ;; +2 = 11C \n\t"
    "_Li2c_write_low: \n\t"
    " sbi %[SDADDR],%[SDAPIN] ;force SDA low       ;; +2 = 9C \n\t"
    " rjmp _Li2c_write_high                  ;;+2 = 11C \n\t"
    "_Li2c_write_high: \n\t"
#if I2C_DELAY_COUNTER >= 1
    " rcall ass_i2c_delay_half ;delay T/2          ;;+X = 11C+X\n\t"
#endif
    " cbi %[SCLDDR],%[SCLPIN] ;release SCL         ;;+2 = 13C+X\n\t"
    " cln                        ;clear N-bit      ;;+1 = 14C+X\n\t"
    " nop \n\t"
    " nop \n\t"
```

```
    " nop \n\t"
    " sbis %[SCLIN],%[SCLPIN] ;check for SCL high    ;;+2 = 16C+X\n\t"
    " rcall    ass_i2c_wait_scl_high \n\t"
    " brpl     _Ldelay_scl_high                      ;;+2 = 18C+X\n\t"
    "_Li2c_write_return_false: \n\t"
    " clr      r24                 ; return false because of timeout \n\t"
    " rjmp     _Li2c_write_return \n\t"
    "_Ldelay_scl_high: \n\t"
#if I2C_DELAY_COUNTER >= 1
    " rcall ass_i2c_delay_half ;delay T/2           ;;+X= 18C+2X\n\t"
#endif
    " rjmp _Li2c_write_bit \n\t"
    "            ;; +2 = 20C +2X for one bit-loop \n\t"
    "_Li2c_get_ack: \n\t"
    " sbi %[SCLDDR],%[SCLPIN] ;force SCL low ;; +2 = 5C \n\t"
    " nop \n\t"
    " nop \n\t"
    " cbi %[SDADDR],%[SDAPIN] ;release SDA ;;+2 = 7C \n\t"
#if I2C_DELAY_COUNTER >= 1
    " rcall ass_i2c_delay_half ;delay T/2 ;; +X = 7C+X \n\t"
#endif
    " clr r25                                   ;; 17C+2X \n\t"
    " clr r24       ;return 0           ;; 14C + X \n\t"
    " cbi %[SCLDDR],%[SCLPIN] ;release SCL ;; +2 = 9C+X\n\t"
    "_Li2c_ack_wait: \n\t"
    " cln                       ; clear N-bit       ;; 10C + X\n\t"
    " nop \n\t"
    " sbis %[SCLIN],%[SCLPIN] ;wait SCL high        ;; 12C + X \n\t"
    " rcall    ass_i2c_wait_scl_high \n\t"
    " brmi     _Li2c_write_return_false                 ;; 13C + X \n\t "
    " sbis %[SDAIN],%[SDAPIN]     ;if SDA hi -> return 0 ;; 15C + X \n\t"
    " ldi r24,1             ;return true        ;; 16C + X \n\t"
#if I2C_DELAY_COUNTER >= 1
    " rcall ass_i2c_delay_half ;delay T/2           ;; 16C + 2X \n\t"
```

```
#endif
    "_Li2c_write_return: \n\t"
    " nop \n\t "
    " nop \n\t "
    " sbi %[SCLDDR],%[SCLPIN] ;force SCL low so SCL=H is short\n\t"
    " ret \n\t"
    "             ;; + 4 = 17C + 2X for acknowldge bit"
    ::
     [SCLDDR] "I"  (SCL_DDR), [SCLPIN] "I" (SCL_PIN), [SCLIN] "I" (SCL_IN),
     [SDADDR] "I"  (SDA_DDR), [SDAPIN] "I" (SDA_PIN), [SDAIN] "I" (SDA_IN));
  return true; // fooling the compiler
}

uint8_t i2c_read(bool last)
{
  __asm__ __volatile__
    (
    " ldi r23,0x01 \n\t"
    "_Li2c_read_bit: \n\t"
    " sbi %[SCLDDR],%[SCLPIN] ;force SCL low        ;; 2C \n\t"
    " cbi %[SDADDR],%[SDAPIN] ;release SDA(prev. ACK);; 4C \n\t"
    " nop \n\t"
    " nop \n\t"
    " nop \n\t"
#if I2C_DELAY_COUNTER >= 1
    " rcall ass_i2c_delay_half ;delay T/2            ;; 4C+X \n\t"
#endif
    " cbi %[SCLDDR],%[SCLPIN] ;release SCL          ;; 6C + X \n\t"
#if I2C_DELAY_COUNTER >= 1
    " rcall ass_i2c_delay_half ;delay T/2            ;; 6C + 2X \n\t"
#endif
    " cln                       ; clear N-bit        ;; 7C + 2X \n\t"
    " nop \n\t "
    " nop \n\t "
```

```
    " nop \n\t "
    " sbis    %[SCLIN], %[SCLPIN]    ;check for SCL high    ;; 9C +2X \n\t"
    " rcall    ass_i2c_wait_scl_high \n\t"
    " brmi    _Li2c_read_return      ;return if timeout    ;; 10C + 2X\n\t"
    " clc        ;clear carry flag     ;; 11C + 2X\n\t"
    " sbic %[SDAIN],%[SDAPIN] ;if SDA is high       ;; 11C + 2X\n\t"
    " sec       ;set carry flag       ;; 12C + 2X\n\t"
    " rol r23       ;store bit           ;; 13C + 2X\n\t"
    " brcc _Li2c_read_bit       ;while receiv reg not full \n\t"
    "                    ;; 15C + 2X for one bit loop \n\t"

    "_Li2c_put_ack: \n\t"
    " sbi %[SCLDDR],%[SCLPIN] ;force SCL low        ;; 2C \n\t"
    " cpi r24,0                                    ;; 3C \n\t"
    " breq _Li2c_put_ack_low ;if (ack=0) ;; 5C \n\t"
    " cbi %[SDADDR],%[SDAPIN] ;release SDA \n\t"
    " rjmp _Li2c_put_ack_high \n\t"
    "_Li2c_put_ack_low:              ;else \n\t"
    " sbi %[SDADDR],%[SDAPIN] ;force SDA low        ;; 7C \n\t"
    "_Li2c_put_ack_high: \n\t"
    " nop \n\t "
    " nop \n\t "
    " nop \n\t "
#if I2C_DELAY_COUNTER >= 1
    " rcall ass_i2c_delay_half ;delay T/2           ;; 7C + X \n\t"
#endif
    " cbi %[SCLDDR],%[SCLPIN] ;release SCL          ;; 9C +X \n\t"
    " cln                          ;clear N             ;; +1 = 10C\n\t"
    " nop \n\t "
    " nop \n\t "
    " sbis %[SCLIN],%[SCLPIN] ;wait SCL high        ;; 12C + X\n\t"
    " rcall    ass_i2c_wait_scl_high \n\t"
#if I2C_DELAY_COUNTER >= 1
    " rcall ass_i2c_delay_half ;delay T/2           ;; 11C + 2X\n\t"
```

```
#endif
    "_Li2c_read_return: \n\t"
    " nop \n\t "
    " nop \n\t "
    "sbi %[SCLDDR],%[SCLPIN] ;force SCL low so SCL=H is short\n\t"
    " mov r24,r23                    ;; 12C + 2X \n\t"
    " clr r25                      ;; 13 C + 2X\n\t"
    " ret                          ;; 17C + X"
    ::
     [SCLDDR] "I"  (SCL_DDR), [SCLPIN] "I" (SCL_PIN), [SCLIN] "I" (SCL_IN),
     [SDADDR] "I"  (SDA_DDR), [SDAPIN] "I" (SDA_PIN), [SDAIN] "I" (SDA_IN)
    );
  return ' '; // fool the compiler!
}

#endif



#endif[/code]
```