

```

#ifndef Pins_Arduino_h
#define Pins_Arduino_h

#include <avr/pgmspace.h>

//                                     ATMEGA1284
//
//                                     +---\ /---+
//      (D 0) PB0 1 |           |40 PA0 (AI 0 / D31)
//      (D 1) PB1 2 |           |39 PA1 (AI 1 / D30)
//      INT2 (D 2) PB2 3 |       |38 PA2 (AI 2 / D29)
//      PWM (D 3) PB3 4 |       |37 PA3 (AI 3 / D28)
//      PWM/SS (D 4) PB4 5 |     |36 PA4 (AI 4 / D27)
//      MOSI (D 5) PB5 6 |     |35 PA5 (AI 5 / D26)
//      PWM/MISO (D 6) PB6 7 |   |34 PA6 (AI 6 / D25)
//      PWM/SCK (D 7) PB7 8 |   |33 PA7 (AI 7 / D24)
//      RST 9 |                 |32 AREF
//      VCC 10 |                |31 GND
//      GND 11 |                |30 AVCC
//      XTAL2 12 |              |29 PC7 (D 23)
//      XTAL1 13 |              |28 PC6 (D 22)
//      RX0 (D 8) PD0 14 |      |27 PC5 (D 21) TDI
//      TX0 (D 9) PD1 15 |      |26 PC4 (D 20) TDO
//      RX1/INT0 (D 10) PD2 16 | |25 PC3 (D 19) TMS
//      TX1/INT1 (D 11) PD3 17 | |24 PC2 (D 18) TCK
//      PWM (D 12) PD4 18 |     |23 PC1 (D 17) SDA
//      PWM (D 13) PD5 19 |     |22 PC0 (D 16) SCL
//      PWM (D 14) PD6 20 |     |21 PD7 (D 15) PWM
//                                     +-----+
//
/*
    PCINT15-8: D7-0    : bit 1
    PCINT31-24: D15-8  : bit 3
    PCINT23-16: D23-16 : bit 2
    PCINT7-0:  D31-24  : bit 0
*/

#define NUM_DIGITAL_PINS          32
#define NUM_ANALOG_INPUTS        8
#define analogInputToDigitalPin(p) ((p < NUM_ANALOG_INPUTS) ? (p) +
24 : -1)
#define digitalPinHasPWM(p)       ((p) == 3 || (p) == 4 || (p) ==
6 || (p) == 7 || (p) == 12 || (p) == 13 || (p) == 14 || (p) == 15)

static const uint8_t SS    = 4;
static const uint8_t MOSI = 5;
static const uint8_t MISO = 6;
static const uint8_t SCK  = 7;

static const uint8_t SDA = 17;
static const uint8_t SCL = 16;

```

```

#define LED_BUILTIN 7

static const uint8_t A0 = 31;
static const uint8_t A1 = 30;
static const uint8_t A2 = 29;
static const uint8_t A3 = 28;
static const uint8_t A4 = 27;
static const uint8_t A5 = 26;
static const uint8_t A6 = 25;
static const uint8_t A7 = 24;

#define digitalPinToPCICR(p)    (((p) >= 0 && (p) <
NUM_DIGITAL_PINS) ? (&PCICR) : ((uint8_t *)0))
#define digitalPinToPCICRbit(p) (((p) <= 7) ? 1 : (((p) <= 15) ? 3 :
(((p) <= 23) ? 2 : 0)))
#define digitalPinToPCMSK(p)    (((p) <= 7) ? (&PCMSK2) : (((p) <=
13) ? (&PCMSK0) : (((p) <= 21) ? (&PCMSK1) : ((uint8_t *)0))))
#define digitalPinToPCMSKbit(p) ((p) % 8)
#define digitalPinToInterrupt(p)    ((p) == 10 ? 0 : ((p) ==
11 ? 1 : ((p) == 2 ? 2 : NOT_AN_INTERRUPT)))

#ifndef ARDUINO_MAIN

const uint16_t PROGMEM port_to_mode_PGM[] =
{
    NOT_A_PORT,
    (uint16_t) &DDRA,
    (uint16_t) &DDRB,
    (uint16_t) &DDRC,
    (uint16_t) &DDRD,
};

const uint16_t PROGMEM port_to_output_PGM[] =
{
    NOT_A_PORT,
    (uint16_t) &PORTA,
    (uint16_t) &PORTB,
    (uint16_t) &PORTC,
    (uint16_t) &PORTD,
};

const uint16_t PROGMEM port_to_input_PGM[] =
{
    NOT_A_PORT,
    (uint16_t) &PINA,
    (uint16_t) &PINB,
    (uint16_t) &PINC,
    (uint16_t) &PIND,
};

const uint8_t PROGMEM digital_pin_to_port_PGM[32] = {
    PB,    // PB0 ** D0
    PB,    // PB1 ** D1
    PB,    // PB2 ** D2

```

```

PB, // PB3 ** D3
PB, // PB4 ** D4
PB, // PB5 ** D5
PB, // PB6 ** D6
PB, // PB7 ** D7
PD, // PD0 ** D8
PD, // PD1 ** D9
PD, // PD2 ** D10
PD, // PD3 ** D11
PD, // PD4 ** D12
PD, // PD5 ** D13
PD, // PD6 ** D14
PD, // PD7 ** D15
PC, // PC0 ** D16
PC, // PC1 ** D17
PC, // PC2 ** D18
PC, // PC3 ** D19
PC, // PC4 ** D20
PC, // PC5 ** D21
PC, // PC6 ** D22
PC, // PC7 ** D23
PA, // PA7 ** A7 D24
PA, // PA6 ** A6 D25
PA, // PA5 ** A5 D26
PA, // PA4 ** A4 D27
PA, // PA3 ** A3 D28
PA, // PA2 ** A2 D29
PA, // PA1 ** A1 D30
PA, // PA0 ** A0 D31

```

```
};
```

```
const uint8_t PROGMEM digital_pin_to_bit_mask_PGM[32] = {
```

```

_BV(0), // PB0 ** D0
_BV(1), // PB1 ** D1
_BV(2), // PB2 ** D2
_BV(3), // PB3 ** D3
_BV(4), // PB4 ** D4
_BV(5), // PB5 ** D5
_BV(6), // PB6 ** D6
_BV(7), // PB7 ** D7
_BV(0), // PD0 ** D8
_BV(1), // PD1 ** D9
_BV(2), // PD2 ** D10
_BV(3), // PD3 ** D11
_BV(4), // PD4 ** D12
_BV(5), // PD5 ** D13
_BV(6), // PD6 ** D14
_BV(7), // PD7 ** D15
_BV(0), // PC0 ** D16
_BV(1), // PC1 ** D17
_BV(2), // PC2 ** D18
_BV(3), // PC3 ** D19
_BV(4), // PC4 ** D20
_BV(5), // PC5 ** D21

```

```

    _BV(6), // PC6 ** D22
    _BV(7), // PC7 ** D23
    _BV(7), // PA7 ** A7 D24
    _BV(6), // PA6 ** A6 D25
    _BV(5), // PA5 ** A5 D26
    _BV(4), // PA4 ** A4 D27
    _BV(3), // PA3 ** A3 D28
    _BV(2), // PA2 ** A2 D29
    _BV(1), // PA1 ** A1 D30
    _BV(0), // PA0 ** A0 D31
};

```

```

const uint8_t PROGMEM digital_pin_to_timer_PGM[] =
{
    NOT_ON_TIMER, /* 0 - PB0 */
    NOT_ON_TIMER, /* 1 - PB1 */
    NOT_ON_TIMER, /* 2 - PB2 */
    TIMER0A,      /* 3 - PB3 */
    TIMER0B,      /* 4 - PB4 */
    NOT_ON_TIMER, /* 5 - PB5 */
    TIMER3A,      /* 6 - PB6 */
    TIMER3B,      /* 7 - PB7 */
    NOT_ON_TIMER, /* 8 - PD0 */
    NOT_ON_TIMER, /* 9 - PD1 */
    NOT_ON_TIMER, /* 10 - PD2 */
    NOT_ON_TIMER, /* 11 - PD3 */
    TIMER1B,      /* 12 - PD4 */
    TIMER1A,      /* 13 - PD5 */
    TIMER2B,      /* 14 - PD6 */
    TIMER2A,      /* 15 - PD7 */
    NOT_ON_TIMER, /* 16 - PC0 */
    NOT_ON_TIMER, /* 17 - PC1 */
    NOT_ON_TIMER, /* 18 - PC2 */
    NOT_ON_TIMER, /* 19 - PC3 */
    NOT_ON_TIMER, /* 20 - PC4 */
    NOT_ON_TIMER, /* 21 - PC5 */
    NOT_ON_TIMER, /* 22 - PC6 */
    NOT_ON_TIMER, /* 23 - PC7 */
    NOT_ON_TIMER, /* 24 - PA7 */
    NOT_ON_TIMER, /* 25 - PA6 */
    NOT_ON_TIMER, /* 26 - PA5 */
    NOT_ON_TIMER, /* 27 - PA4 */
    NOT_ON_TIMER, /* 28 - PA3 */
    NOT_ON_TIMER, /* 29 - PA2 */
    NOT_ON_TIMER, /* 30 - PA1 */
    NOT_ON_TIMER /* 31 - PA0 */
};

```

```

#endif
// These serial port names are intended to allow libraries and
// architecture-neutral
// sketches to automatically default to the correct port name for a
// particular type
// of use. For example, a GPS module would normally connect to

```

```
SERIAL_PORT_HARDWARE_OPEN,  
// the first hardware serial port whose RX/TX pins are not dedicated  
// to another use.  
//  
// SERIAL_PORT_MONITOR          Port which normally prints to the  
// Arduino Serial Monitor  
//  
// SERIAL_PORT_USBVIRTUAL       Port which is USB virtual serial  
//  
// SERIAL_PORT_LINUXBRIDGE      Port which connects to a Linux system  
// via Bridge library  
//  
// SERIAL_PORT_HARDWARE         Hardware serial port, physical RX & TX  
// pins.  
//  
// SERIAL_PORT_HARDWARE_OPEN     Hardware serial ports which are open  
// for use. Their RX & TX  
// pins are NOT connected to anything by  
// default.  
#define SERIAL_PORT_MONITOR     Serial  
#define SERIAL_PORT_HARDWARE    Serial  
  
#endif
```