#ifndef Adafruit\_BusIO\_Register\_h

#define Adafruit\_BusIO\_Register\_h

#include <Arduino.h>

#if !defined(SPI\_INTERFACES\_COUNT) || \

(defined(SPI\_INTERFACES\_COUNT) && (SPI\_INTERFACES\_COUNT > 0))

#include <Adafruit\_I2CDevice.h>

#include <Adafruit\_SPIDevice.h>

typedef enum \_Adafruit\_BusIO\_SPIRegType {

ADDRBIT8\_HIGH\_TOREAD = 0,

/\*!<

\* ADDRBIT8\_HIGH\_TOREAD

\* When reading a register you must actually send the value 0x80 + register

\* address to the device. e.g. To read the register 0x0B the register value

\* 0x8B is sent and to write 0x0B is sent.

\*/

AD8\_HIGH\_TOREAD\_AD7\_HIGH\_TOINC = 1,

/\*!<

\* ADDRBIT8\_HIGH\_TOWRITE

\* When writing to a register you must actually send the value 0x80 +

\* the register address to the device. e.g. To write to the register 0x19 the

\* register value 0x99 is sent and to read 0x19 is sent.

\*/

ADDRBIT8\_HIGH\_TOWRITE = 2,

/\*!<

\* ADDRESSED\_OPCODE\_LOWBIT\_TO\_WRITE

\* Used by the MCP23S series, we send 0x40 |'rd with the opcode

\* Then set the lowest bit to write

\*/

ADDRESSED\_OPCODE\_BIT0\_LOW\_TO\_WRITE = 3,

} Adafruit\_BusIO\_SPIRegType;

/\*!

\* @brief The class which defines a device register (a location to read/write

\* data from)

\*/

class Adafruit\_BusIO\_Register {

public:

Adafruit\_BusIO\_Register(Adafruit\_I2CDevice \*i2cdevice, uint16\_t reg\_addr,

uint8\_t width = 1, uint8\_t byteorder = LSBFIRST,

uint8\_t address\_width = 1);

Adafruit\_BusIO\_Register(Adafruit\_SPIDevice \*spidevice, uint16\_t reg\_addr,

Adafruit\_BusIO\_SPIRegType type, uint8\_t width = 1,

uint8\_t byteorder = LSBFIRST,

uint8\_t address\_width = 1);

Adafruit\_BusIO\_Register(Adafruit\_I2CDevice \*i2cdevice,

Adafruit\_SPIDevice \*spidevice,

Adafruit\_BusIO\_SPIRegType type, uint16\_t reg\_addr,

uint8\_t width = 1, uint8\_t byteorder = LSBFIRST,

uint8\_t address\_width = 1);

bool read(uint8\_t \*buffer, uint8\_t len);

bool read(uint8\_t \*value);

bool read(uint16\_t \*value);

uint32\_t read(void);

uint32\_t readCached(void);

bool write(uint8\_t \*buffer, uint8\_t len);

bool write(uint32\_t value, uint8\_t numbytes = 0);

uint8\_t width(void);

void setWidth(uint8\_t width);

void setAddress(uint16\_t address);

void setAddressWidth(uint16\_t address\_width);

void print(Stream \*s = &Serial);

void println(Stream \*s = &Serial);

private:

Adafruit\_I2CDevice \*\_i2cdevice;

Adafruit\_SPIDevice \*\_spidevice;

Adafruit\_BusIO\_SPIRegType \_spiregtype;

uint16\_t \_address;

uint8\_t \_width, \_addrwidth, \_byteorder;

uint8\_t \_buffer[4]; // we won't support anything larger than uint32 for

// non-buffered read

uint32\_t \_cached = 0;

};

/\*!

\* @brief The class which defines a slice of bits from within a device register

\* (a location to read/write data from)

\*/

class Adafruit\_BusIO\_RegisterBits {

public:

Adafruit\_BusIO\_RegisterBits(Adafruit\_BusIO\_Register \*reg, uint8\_t bits,

uint8\_t shift);

bool write(uint32\_t value);

uint32\_t read(void);

private:

Adafruit\_BusIO\_Register \*\_register;

uint8\_t \_bits, \_shift;

};

#endif // SPI exists

#endif // BusIO\_Register\_h