#ifndef Adafruit\_SPIDevice\_h

#define Adafruit\_SPIDevice\_h

#include <Arduino.h>

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

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

// HW SPI available

#include <SPI.h>

#define BUSIO\_HAS\_HW\_SPI

#else

// SW SPI ONLY

enum { SPI\_MODE0, SPI\_MODE1, SPI\_MODE2, \_SPI\_MODE4 };

typedef uint8\_t SPIClass;

#endif

// some modern SPI definitions don't have BitOrder enum

#if (defined(\_\_AVR\_\_) && !defined(ARDUINO\_ARCH\_MEGAAVR)) || \

defined(ESP8266) || defined(TEENSYDUINO) || defined(SPARK) || \

defined(ARDUINO\_ARCH\_SPRESENSE) || defined(MEGATINYCORE) || \

defined(DXCORE) || defined(ARDUINO\_AVR\_ATmega4809) || \

defined(ARDUINO\_AVR\_ATmega4808) || defined(ARDUINO\_AVR\_ATmega3209) || \

defined(ARDUINO\_AVR\_ATmega3208) || defined(ARDUINO\_AVR\_ATmega1609) || \

defined(ARDUINO\_AVR\_ATmega1608) || defined(ARDUINO\_AVR\_ATmega809) || \

defined(ARDUINO\_AVR\_ATmega808) || defined(ARDUINO\_ARCH\_ARC32) || \

defined(ARDUINO\_ARCH\_XMC)

typedef enum \_BitOrder {

SPI\_BITORDER\_MSBFIRST = MSBFIRST,

SPI\_BITORDER\_LSBFIRST = LSBFIRST,

} BusIOBitOrder;

#elif defined(ESP32) || defined(\_\_ASR6501\_\_) || defined(\_\_ASR6502\_\_)

// some modern SPI definitions don't have BitOrder enum and have different SPI

// mode defines

typedef enum \_BitOrder {

SPI\_BITORDER\_MSBFIRST = SPI\_MSBFIRST,

SPI\_BITORDER\_LSBFIRST = SPI\_LSBFIRST,

} BusIOBitOrder;

#else

// Some platforms have a BitOrder enum but its named MSBFIRST/LSBFIRST

#define SPI\_BITORDER\_MSBFIRST MSBFIRST

#define SPI\_BITORDER\_LSBFIRST LSBFIRST

typedef BitOrder BusIOBitOrder;

#endif

#if defined(\_\_IMXRT1062\_\_) // Teensy 4.x

// \*Warning\* I disabled the usage of FAST\_PINIO as the set/clear operations

// used in the cpp file are not atomic and can effect multiple IO pins

// and if an interrupt happens in between the time the code reads the register

// and writes out the updated value, that changes one or more other IO pins

// on that same IO port, those change will be clobbered when the updated

// values are written back. A fast version can be implemented that uses the

// ports set and clear registers which are atomic.

// typedef volatile uint32\_t BusIO\_PortReg;

// typedef uint32\_t BusIO\_PortMask;

//#define BUSIO\_USE\_FAST\_PINIO

#elif defined(ARDUINO\_ARCH\_XMC)

#undef BUSIO\_USE\_FAST\_PINIO

#elif defined(\_\_AVR\_\_) || defined(TEENSYDUINO)

typedef volatile uint8\_t BusIO\_PortReg;

typedef uint8\_t BusIO\_PortMask;

#define BUSIO\_USE\_FAST\_PINIO

#elif defined(ESP8266) || defined(ESP32) || defined(\_\_SAM3X8E\_\_) || \

defined(ARDUINO\_ARCH\_SAMD)

typedef volatile uint32\_t BusIO\_PortReg;

typedef uint32\_t BusIO\_PortMask;

#define BUSIO\_USE\_FAST\_PINIO

#elif (defined(\_\_arm\_\_) || defined(ARDUINO\_FEATHER52)) && \

!defined(ARDUINO\_ARCH\_MBED) && !defined(ARDUINO\_ARCH\_RP2040) && \

!defined(ARDUINO\_SILABS) && !defined(ARDUINO\_UNOR4\_MINIMA) && \

!defined(ARDUINO\_UNOR4\_WIFI)

typedef volatile uint32\_t BusIO\_PortReg;

typedef uint32\_t BusIO\_PortMask;

#if !defined(\_\_ASR6501\_\_) && !defined(\_\_ASR6502\_\_)

#define BUSIO\_USE\_FAST\_PINIO

#endif

#else

#undef BUSIO\_USE\_FAST\_PINIO

#endif

/\*\*! The class which defines how we will talk to this device over SPI \*\*/

class Adafruit\_SPIDevice {

public:

#ifdef BUSIO\_HAS\_HW\_SPI

Adafruit\_SPIDevice(int8\_t cspin, uint32\_t freq = 1000000,

BusIOBitOrder dataOrder = SPI\_BITORDER\_MSBFIRST,

uint8\_t dataMode = SPI\_MODE0, SPIClass \*theSPI = &SPI);

#else

Adafruit\_SPIDevice(int8\_t cspin, uint32\_t freq = 1000000,

BusIOBitOrder dataOrder = SPI\_BITORDER\_MSBFIRST,

uint8\_t dataMode = SPI\_MODE0, SPIClass \*theSPI = nullptr);

#endif

Adafruit\_SPIDevice(int8\_t cspin, int8\_t sck, int8\_t miso, int8\_t mosi,

uint32\_t freq = 1000000,

BusIOBitOrder dataOrder = SPI\_BITORDER\_MSBFIRST,

uint8\_t dataMode = SPI\_MODE0);

~Adafruit\_SPIDevice();

bool begin(void);

bool read(uint8\_t \*buffer, size\_t len, uint8\_t sendvalue = 0xFF);

bool write(const uint8\_t \*buffer, size\_t len,

const uint8\_t \*prefix\_buffer = nullptr, size\_t prefix\_len = 0);

bool write\_then\_read(const uint8\_t \*write\_buffer, size\_t write\_len,

uint8\_t \*read\_buffer, size\_t read\_len,

uint8\_t sendvalue = 0xFF);

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

uint8\_t transfer(uint8\_t send);

void transfer(uint8\_t \*buffer, size\_t len);

void beginTransaction(void);

void endTransaction(void);

void beginTransactionWithAssertingCS();

void endTransactionWithDeassertingCS();

private:

#ifdef BUSIO\_HAS\_HW\_SPI

SPIClass \*\_spi = nullptr;

SPISettings \*\_spiSetting = nullptr;

#else

uint8\_t \*\_spi = nullptr;

uint8\_t \*\_spiSetting = nullptr;

#endif

uint32\_t \_freq;

BusIOBitOrder \_dataOrder;

uint8\_t \_dataMode;

void setChipSelect(int value);

int8\_t \_cs, \_sck, \_mosi, \_miso;

#ifdef BUSIO\_USE\_FAST\_PINIO

BusIO\_PortReg \*mosiPort, \*clkPort, \*misoPort, \*csPort;

BusIO\_PortMask mosiPinMask, misoPinMask, clkPinMask, csPinMask;

#endif

bool \_begun;

};

#endif // Adafruit\_SPIDevice\_h