Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

MAXREFDES72#: Quick Start Guide

Required Equipment

- Device with a modern web browser and the ability to write to an external USB drive
- Microcontroller board with Arduino® UNO R3-compatible socket headers
- MAXREFDES72# board
- MAX3232PMB1 or other Pmod™-compatible peripheral module, available here

Overview

This Quick Start Guide explains how to configure the MAXREFDES72# to map functions from Arduino headers to a Pmod connector. The MAXREFDES72# is capable of arbitrary pin mapping with a few restrictions. In general, the board is configured by writing to the MAX14661 multiplexer over the I^2C bus. As the Figure 2 shows, 8 Arduino digital GPIO are connected to the Pmod connector through level shifters and each of these 8 signals are also connected to the multiplexer. The 8 digital signals were chosen so that the Arduino Uno R3 SPI signals are connected for the Pmod Type 2 and to avoid the I^2C and UART signals at the Arduino headers. For Type 1 (GPIO) and Type 2 (SPI), no configuration is necessary. To implement Type 3, Type 4, I^2C, or other custom pin mappings that need access to I^2C, UART, or other real time functions such as PWM, configure the multiplexer to route the signals to the appropriate pins.

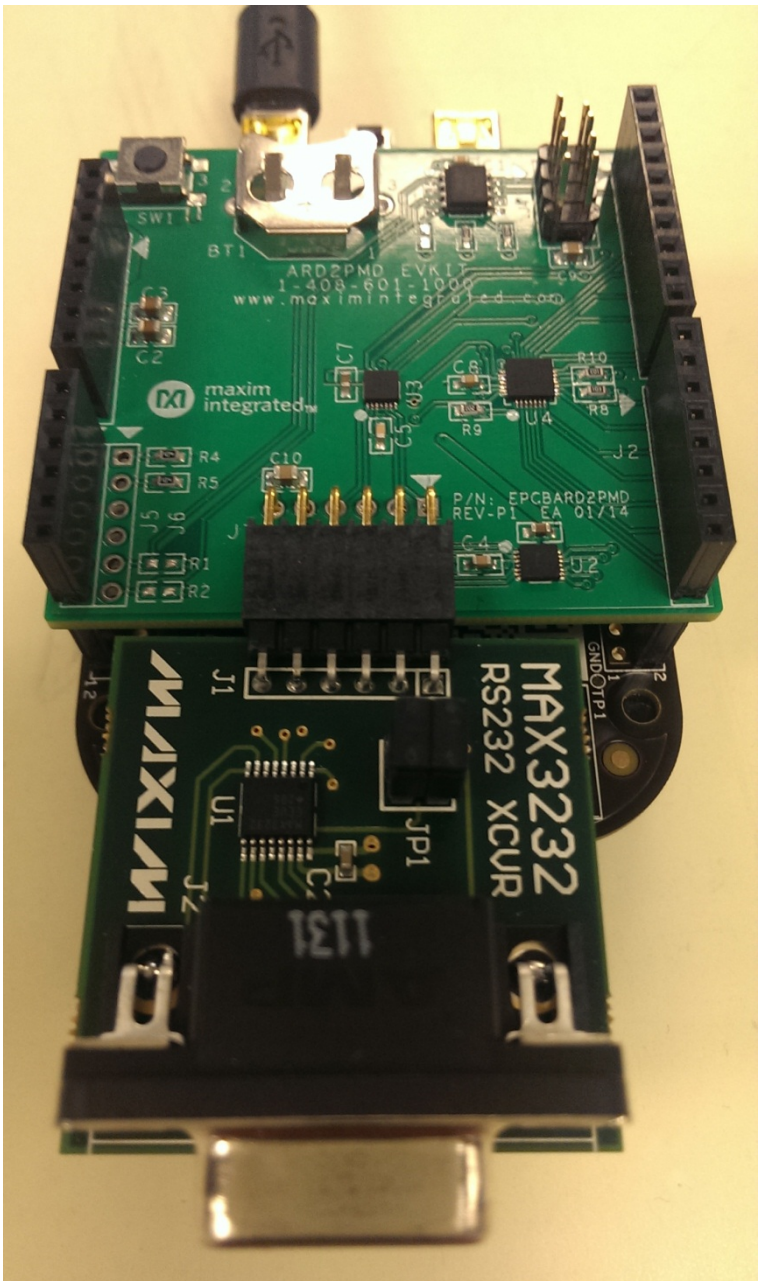


Figure 1

maxim integrated™

Maxim Integrated 160 Rio Robles, San Jose, CA 95134 USA 1-408-601-1000

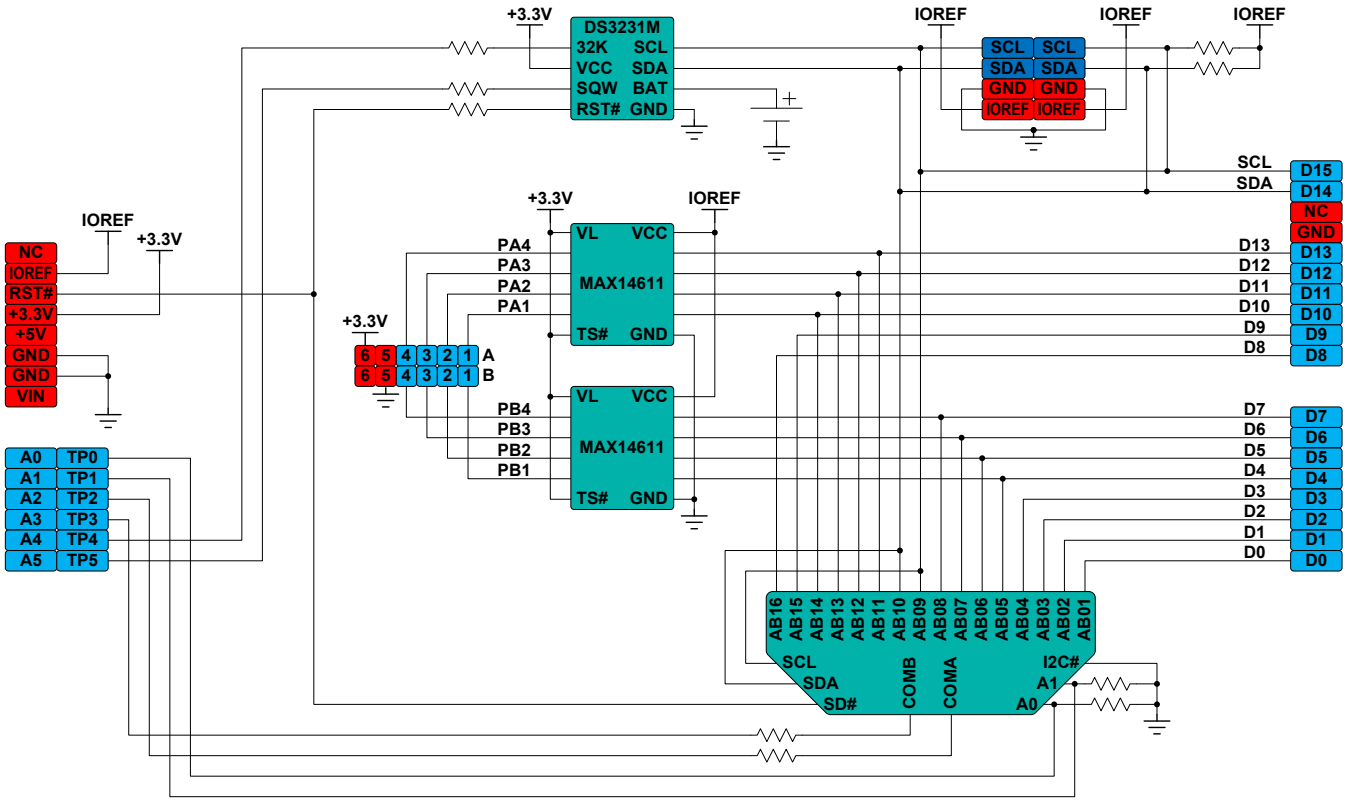


Figure 2

The MAX14661 is a 16:2 matrix multiplexer with all 16 of the Arduino digital IO connected to the 16 port side of the matrix switch. The COM pins on the 2 port side of the matrix switch are used internally for routing, but are also connected to test points and can be used for debug. To route the UART to the Pmod connector, turn on the A switch connected to the RX (AB01) and the A switch connected to the desired Pmod pin (AB12), then turn on the B switch connected to TX (AB02) and the B switch connected to the desired Pmod pin (AB13). Any of the 8 pins that are not connected to the Pmod connector directly can be connected to any of the 8 Pmod connector pins and the only limitation is that only two independent signals can be routed like this at the same time. The multiplexer is programmable and the two channels can be time multiplexed if necessary. An example demonstrating how to do this with the mbed platform is described in the *Included Files* and *Procedure* sections.

In the included example, the microcontroller is only used to configure the MAX14661. The RS232 transceiver is actually being connected to the UART that is the virtual com port on the USB interface. The UART inside the microcontroller needs to be disabled. The flexibility of the MAXREFDES72# adapter allows the Rx and Tx to be swapped to connect the Pmod to the UART inside the microcontroller or to the USB virtual com port. Be careful to check how these signals are connected on the platform you have selected. See *Appendix C: mbed Enabled Board Compatibility* for some of the known compatibility issues.

Included Files

- MAX14661.cpp
- MAX14661.h
- MAX3232_DEMO.cpp

Procedure

This is an example of how to configure the multiplexer using the mbed platform.

1. Connect hardware as shown in Figure 1.
2. Select an mbed enabled board with Arduino headers. See *Appendix C: mbed Enabled Board Compatibility* for known compatibility issues.
3. From the platform page for the selected platform, click **Open mbed Compiler**. You might need to click **Add to your mbed Compiler** first if you have not already added it.
4. A dialogue box will open, asking you to create new program. Check the platform is the one you just selected, choose **Empty Program** for the template, and pick a program name (MAX3232_DEMO).
5. With your new empty program selected in the compiler, click New >> New File and name the file **main.cpp** in the dialogue box.
6. Cut and paste the MAX3232_DEMO code from A*ppendix D: MAX3232 Demonstration Code* into the blank main.cpp file.
7. Click the **Import** button at the top of the compiler.
8. Click to highlight the top item in the list of results (**mbed**), then click **Import** in the upper right of the compile window, and click the **Import** button at the bottom of the dialogue box that pops up.
9. Click the **Import** button at the top of the compiler again.
10. Search for **MAX14661**, click to highlight it in the list of results, then click **Import** in the upper right of the compile window, and click the **Import** button at the bottom of the dialogue box that pops up.
11. With your new program highlighted in the Program Workspace, click the **Compile** button at the top of the compile window and save the file to your computer.
12. Plug the MAXREFDES72# board into your mbed enabled board.
13. Plug the MAX3232PMB1 into the top row of the Pmod connector on the MAXREFDES72# board.
14. Connect the mbed interface USB connector to your computer with a USB cable.
15. Drag and drop the .bin file to the mbed drive that appears when the board is plugged into USB.
16. Press the reset button and use your USB to RS232 adapter. Refer to the "Windows serial configuration" page to download the driver needed by Windows®.

Appendix A: Multiplexer Connections

MAX14661	Arduino	Pmod
SW01	D0	
SW02	D1	
SW03	D2	
SW04	D3	
SW05	D4	PB1
SW06	D5	PB2
SW07	D6	PB3
SW08	D7	PB4
SW09	D15 (SCL)	
SW10	D14 (SDA)	
SW11	D13	PA4
SW12	D12	PA3
SW13	D11	PA2
SW14	D10	PA1
SW15	D9	
SW16	D8	

Arduino	MAX14661	Pmod
D0	SW01	
D1	SW02	
D2	SW03	
D3	SW04	
D4	SW05	PB1
D5	SW06	PB2
D6	SW07	PB3
D7	SW08	PB4
D8	SW16	
D9	SW15	
D10	SW14	PA1
D11	SW13	PA2
D12	SW12	PA3
D13	SW11	PA4
D14 (SDA)	SW10	
D15 (SCL)	SW09	

Appendix B: Pmod Pin Mapping

Pin	PMOD	I^2C	Type 1 GPIO	Type 2 SPI	Type 3 UART	Type 4 UART	Type 5 H-Bridge	Type 6 Dual H-Bridge
A1	**1**		IO1	SS	CTS	CTS	DIR	DIR1
A2	**2**		IO2	MOSI	RTS	TXD	EN	EN1
A3	**3**	SCL	IO3	MISO	RXD	RXD	SA	DIR2
A4	**4**	SDA	IO4	SCK	TXD	RTS	SB	EN2
A5	**5**	GND	GND	GND	GND	GND	GND	GND
A6	**6**	VCC	VCC	VCC	VCC	VCC	VCC	VCC
B1	**7**			INT		INT		
B2	**8**			RESET		RESET		
B3	**9**	SCL		N/S		N/S		
B4	**10**	SDA		N/S		N/S		
B5	**11**	GND	GND	GND	GND	GND	GND	GND
B6	**12**	VCC	VCC	VCC	VCC	VCC	VCC	VCC

Appendix C: mbed Enabled Board Compatibility

This is a summary of known compatibility issues with some of the mbed Enabled Boards. The MAXREFDES72# can work with most boards that have Arduino Uno R3 compatible pin headers, but due to the variability of each implementation the compatibility must be reviewed carefully. This is not an exhaustive list of compatibility issues, but is provided for guidance.

Seeeduino Arch

- Revision 1.0 has the SPI SCK and MOSI pins swapped with respect to Arduino Uno. This has been corrected in revision 1.1.

- This board does not have a dedicated mbed interface chip, but uses the USB bootloader in the LPC11U24 ROM. The lack of a dedicated mbed interface chip means USBRX and USBTX are not present and the firmware update procedure is different. This prevents the included MAX3232 demonstration code from functioning with this board.
- PWM function is not available on D3.

Seeeduino Arch Pro

- PWM function is not available on D3.

FRDM-KL05Z

- I^2C and D4 are shared with MMA8451(0011101).

FRDM-KL25Z

- D13 is pulled up by an LED.

FRDM-KL46Z

- D13 is pulled up by an LED.

FRDM-K64F

- I^2C and D4 are shared with FXOS8700 (0011101).
- mbed USB serial port does not connect to D0, D1. This prevents the included MAX3232 demonstration code from functioning with this board.

STM32 Nucleo

- Pins D0 and D1 are unconnected by default. Need to short missing solder bridges to connect. This modification is required for the included MAX3232 demonstration code to function properly.
- D13 is pulled down by an LED. This makes it difficult to use the Pmod connector top row (A) for I^2C.
- F030R8 version does not have PWM function at D3.

Appendix D: MAX3232 Demonstration Code

```
#include "mbed.h"
#include "MAX14661.h"

MAX14661 mux(D14, D15);
DigitalOut pinRTS(D13);
DigitalIn pinRX(D0);  // Set as input to remove load from mbedTX
DigitalIn pinTX(D1);  // Set as input to remove load from mbedRX

int main()
{
    pinRTS=0; // Not Ready to Send
    // D0 (mbedTX) + PA2 (TXD),  D1 (mbedRX) + PA3 (RXD)
    mux.setAB((MAX14661::SW01 | MAX14661::SW13),
              (MAX14661::SW02 | MAX14661::SW12));
    pinRTS=1; // Ready to Send
}
```

Appendix E: MAX14661 Library

MAX14661.h

```
#ifndef MAX14661_H
#define MAX14661_H

#include "mbed.h"

/** MAX14661 Library, Provides utilities for configuring the MAX14661
over I2C
 *
 * Example:
 * @code
 * // Enable only switch B3 and read back switch state.
 *
 * #include "MAX14661.h"
 *
 * MAX14661 mux(p28, p27);
 *
 * int main() {
 *     mux.setAB(0x0000, MAX14661::SW03);
 *     printf("mux = 0x%08X\n", mux.read());
 *     mux.clearAll();
 *     printf("mux = 0x%08X\n", mux.read());
 * }
 * @endcode
 */
class MAX14661
{
public:

    /** Create a MAX14661 interface
     *
     * @param sda I2C data line pin
     * @param scl I2C clock line pin
     * @param addr MAX14661 I2C address
     */
    MAX14661(PinName sda, PinName scl, int addr = 0x98);

    ~MAX14661();

    /** Name the register addresses
    */
    enum MAX14661regs {
        REG_DIR0 = 0x00,  /**< 8A-1A Direct Access */
        REG_DIR1,         /**< 16A-9A Direct Access */
        REG_DIR2,         /**< 8B-1B Direct Access */
        REG_DIR3,         /**< 16B-9B Direct Access */
        REG_SHDW0 = 0x10, /**< 8A-1A Shadow */
        REG_SHDW1,        /**< 16A-9A Shadow */
        REG_SHDW2,        /**< 8B-1B Shadow */
        REG_SHSW3,        /**< 16B-9B Shadow */
        REG_CMD_A,        /**< Command A */
        REG_CMD_B         /**< Command A */
    };
```

```
/** Name the command codes
*/
enum MAX14661cmds {
    CMD_EN01 = 0x00,  /**< Enable switch 1 */
    CMD_EN02,         /**< Enable switch 2 */
    CMD_EN03,         /**< Enable switch 3 */
    CMD_EN04,         /**< Enable switch 4 */
    CMD_EN05,         /**< Enable switch 5 */
    CMD_EN06,         /**< Enable switch 6 */
    CMD_EN07,         /**< Enable switch 7 */
    CMD_EN08,         /**< Enable switch 8 */
    CMD_EN09,         /**< Enable switch 9 */
    CMD_EN10,         /**< Enable switch 10 */
    CMD_EN11,         /**< Enable switch 11 */
    CMD_EN12,         /**< Enable switch 12 */
    CMD_EN13,         /**< Enable switch 13 */
    CMD_EN14,         /**< Enable switch 14 */
    CMD_EN15,         /**< Enable switch 15 */
    CMD_EN16,         /**< Enable switch 16 */
    CMD_DIS,          /**< Disable switches  */
    CMD_COPY,         /**< Copy shadow registers to switches */
    CMD_NOOP = 0x1F   /**< Keep current state, no changes */
};

/** Name the switch bits
*/
enum MAX14661sws {
    SW01 = (1 << 0),  /**< Bit mask for switch 1 */
    SW02 = (1 << 1),  /**< Bit mask for switch 2 */
    SW03 = (1 << 2),  /**< Bit mask for switch 3 */
    SW04 = (1 << 3),  /**< Bit mask for switch 4 */
    SW05 = (1 << 4),  /**< Bit mask for switch 5 */
    SW06 = (1 << 5),  /**< Bit mask for switch 6 */
    SW07 = (1 << 6),  /**< Bit mask for switch 7 */
    SW08 = (1 << 7),  /**< Bit mask for switch 8 */
    SW09 = (1 << 8),  /**< Bit mask for switch 9 */
    SW10 = (1 << 9),  /**< Bit mask for switch 10 */
    SW11 = (1 << 10), /**< Bit mask for switch 11 */
    SW12 = (1 << 11), /**< Bit mask for switch 12 */
    SW13 = (1 << 12), /**< Bit mask for switch 13 */
    SW14 = (1 << 13), /**< Bit mask for switch 14 */
    SW15 = (1 << 14), /**< Bit mask for switch 15 */
    SW16 = (1 << 15)  /**< Bit mask for switch 16 */
};

/** Clears all bits to opens all 32 switches
 */
void clearAll();

/** Set all 32 switches simultaneously
 *
 *  @param swA the desired state of switches [A16 - A01]
 *  @param swB the desired state of switches [B16 - B01]
 */
void setAB(int swA, int swB);
```

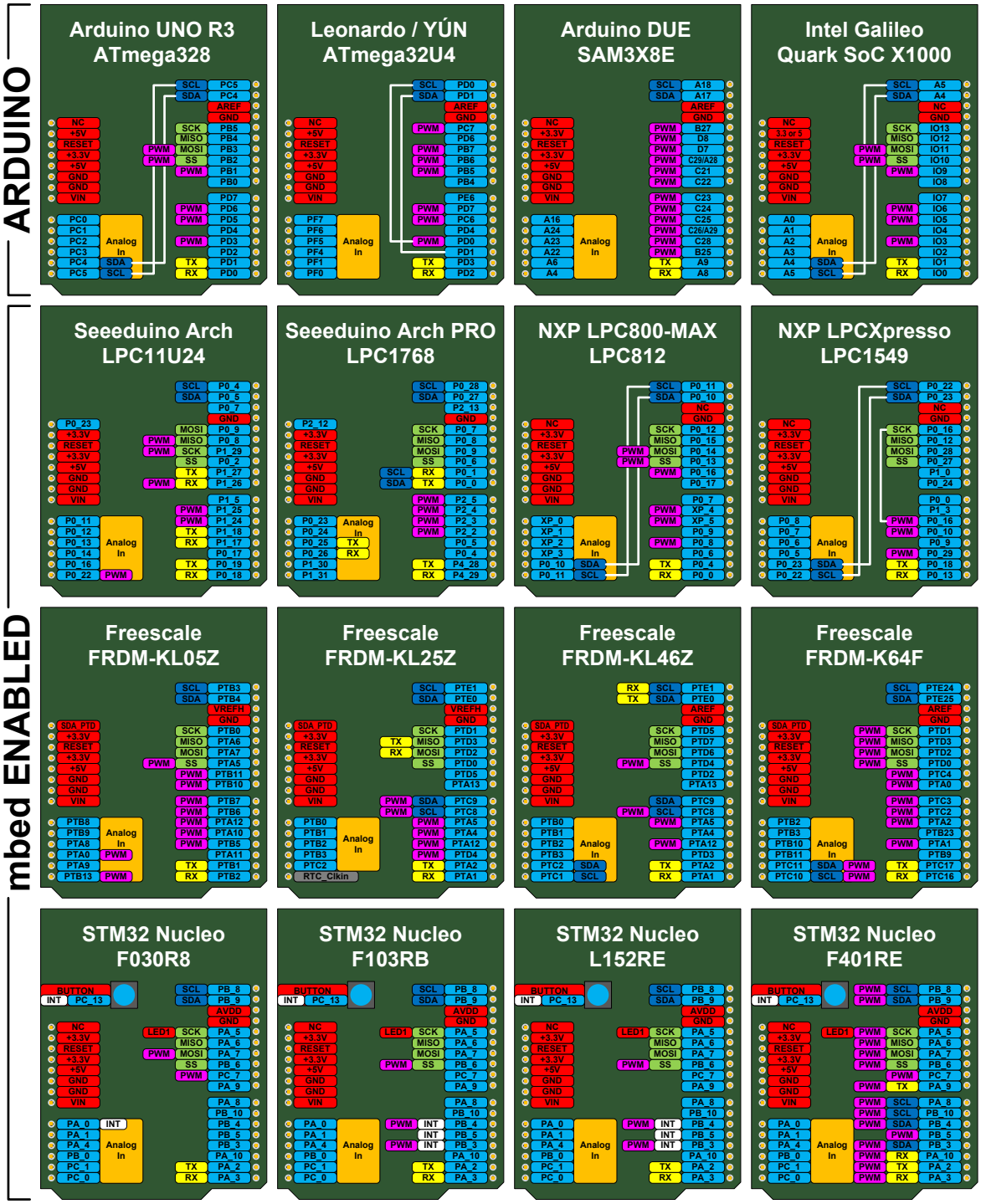
```
    /** Read the status of all 32 switches concatenated into a single
int
     *
     *  @returns
     *    the switch states [B16-B01,A16-A1]
     */
    int read();

private:
    I2C _i2c;
    int _addr;
};

#endif
```

MAX14661.cpp

```
#include "MAX14661.h"
#include "mbed.h"

MAX14661::MAX14661(PinName sda, PinName scl, int addr) : _i2c(sda,
scl)
{
    _addr = addr;
}

MAX14661::~MAX14661()
{
}

void MAX14661::clearAll()
{
    char data[3];
    data[0] = REG_CMD_A;
    data[1] = CMD_DIS;
    data[2] = CMD_DIS;
    _i2c.write(_addr, data, 3);
}

void MAX14661::setAB(int swA, int swB)
{
    char data[7];
    data[0] = REG_SHDW0;
    data[1] = swA;
    data[2] = swA >> 8;
    data[3] = swB;
    data[4] = swB >> 8;
    data[5] = CMD_COPY;
    data[6] = CMD_COPY;
    _i2c.write(_addr, data, 7);
}

int MAX14661::read()
{
    char data[4];
    data[0] = REG_DIR0;
```

```
        _i2c.write(_addr, data, 1, true);
        _i2c.read(_addr, data, 4);
        return ((data[3] << 24) | (data[2] << 16) |(data[1] << 8) |
data[0]);
}
```

Appendix F: Arduino Variant Pin Mapping



Trademarks

Arduino is a registered trademark of Arduino, LLC.
Digilent is a registered trademark and Pmod™ is a trademark of Digilent Inc.
Pmod is a trademark of Diligent Inc.
Windows is a registered trademark and registered service mark of Microsoft Corporation.

Revision History

REV NUMBER	DATE	DESCRIPTION	PAGES CHANGED
0	7/14	Initial release	—