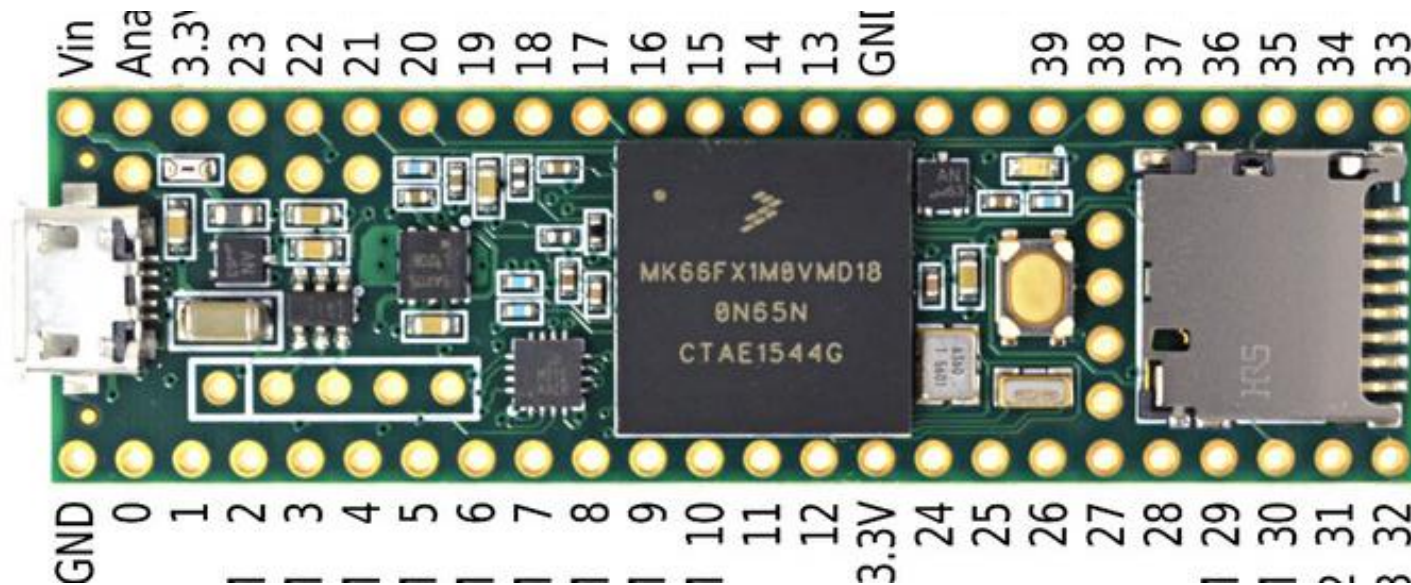


Manipulating Pins on the Teensy 3.5



Data Types

- short, int, long: size depends on the particular microprocessor
- In order to be clear about sizes, gcc (our compiler) provides a set of types, including:
 - int8_t 8-bit signed
 - uint16_t 16-bit unsigned
 - uint32_t 32-bit unsigned
- Use these for our projects – not short, int, long

Teensy 3.5

				GND			Vin (3.6 to 6.0 volts)
Touch	MOSI1	RX1	0				Analog GND
Touch	MISO1	TX1	1				3.3V (250 mA max)
			PWM	2		23 A9 PWM	Touch
SCL2	CAN0TX		PWM	3		22 A8 PWM	Touch
SDA2	CAN0RX		PWM	4		21 A7 PWM	CS0 mosi1
	miso1	tx1	PWM	5		20 A6 PWM	CS0 sck1
			PWM	6		19 A5	SCL0 Touch
scl0	mosi0	RX3	PWM	7		18 A4	SDA0 Touch
sda0	miso0	TX3	PWM	8		17 A3	sda0 Touch
	CS0	RX2	PWM	9		16 A2	scl0 Touch
	CS0	TX2	PWM	10		15 A1	CS0 Touch
	MOSI0			11		14 A0 PWM	sck0
	MISO0			12		13 (LED)	SCK0
			3.3V			GND	
			24			A22 DAC1	
			25			A21 DAC0	
		tx1	26			39 A20	
		rx1	27			38 A19 PWM	SDA1
			28			37 A18 PWM	SCL1
Touch	can0tx		PWM	29		36 A17 PWM	
Touch	can0rx		PWM	30		35 A16 PWM	
	CS1	RX4	A12	31		34 A15 CAN1RX	sda0
	SCK1	TX4	A13	32		33 A14 CAN1TX	scl0

Teensy 3.5

- Floating Point Unit (FPU): high-speed math
- Serial I/O: RS232, I2C, SPI, CAN, Ethernet
- Digital I/O
- Pulse Width Modulation (PWM)
- Multiple timers
- Digital-to-analog converter channels (2)
- Analog-to-digital converter channels (25)

Digital Input/Output

The Teensy encodes a digital value using 0V (low) and 3.3V (high)

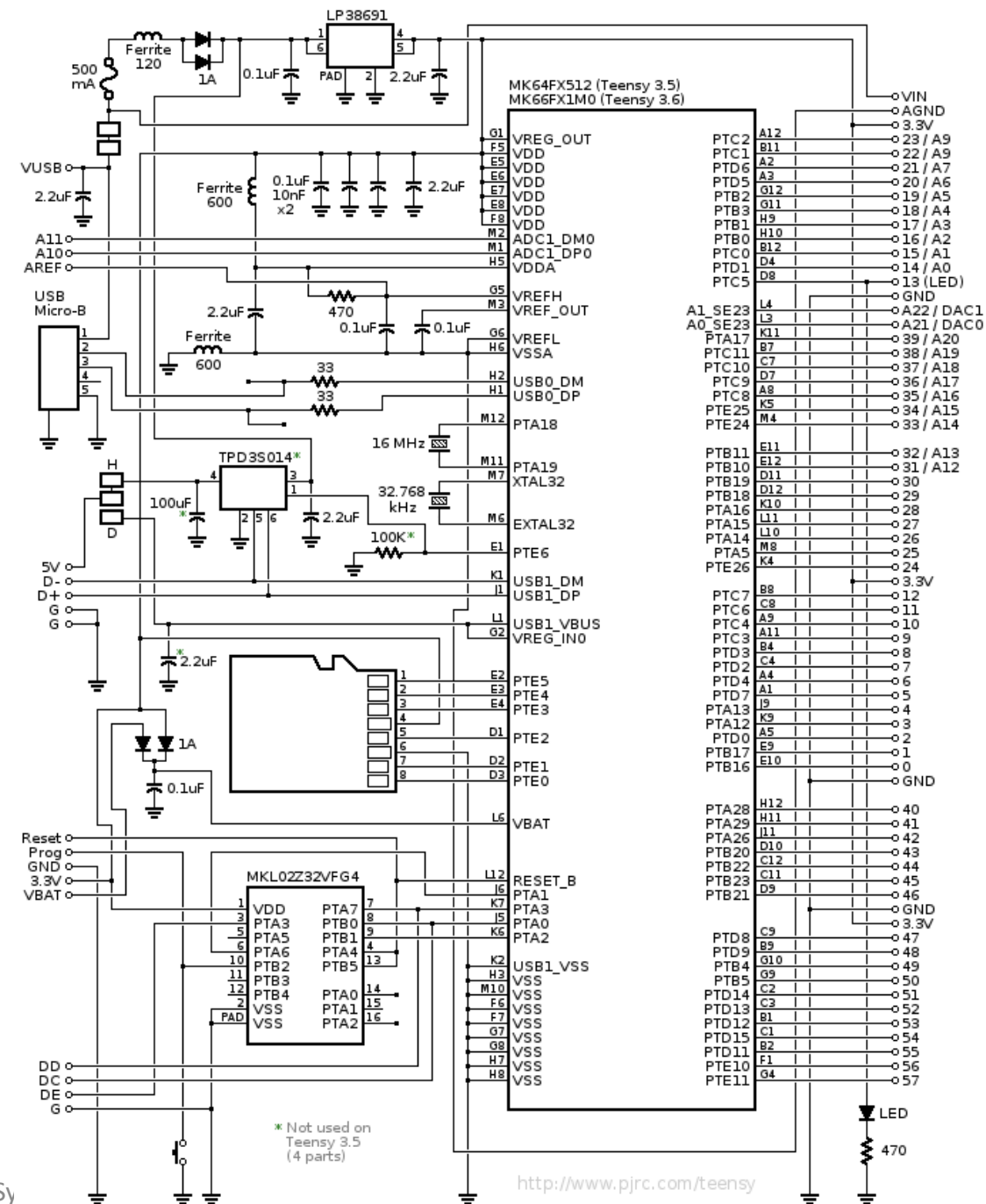
- If a pin is an input:
 - We can ask the pin what its voltage state is
 - Possible answers: 0 or 1 (low or high)
- If a pin is an output:
 - We can drive the pin to be 0V or 3.3V
 - Again, these are encoded digitally as 0 or 1

Digital Input/Output

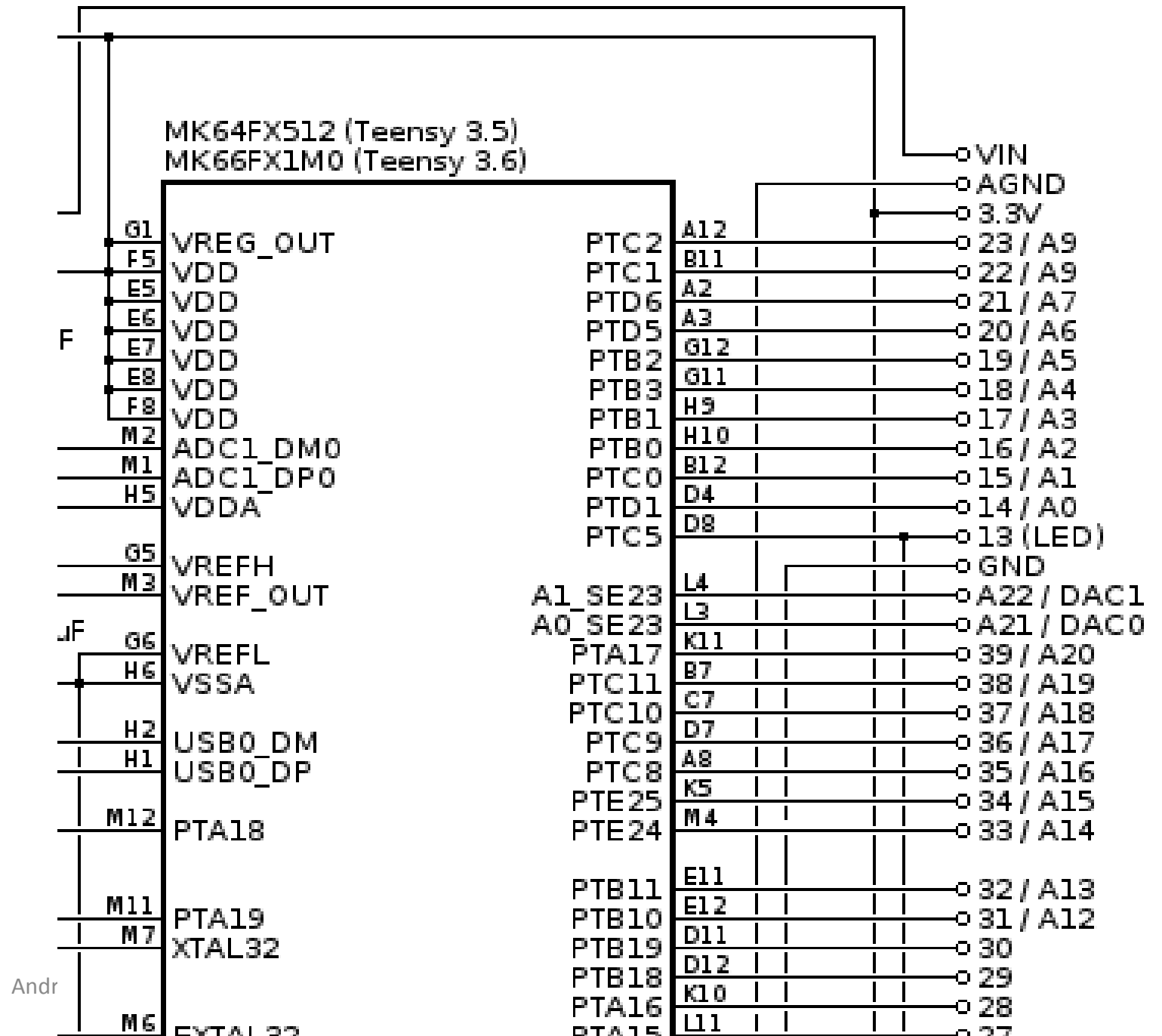
- Pins are organized into groups, called *PORTS*
- Each port can be composed of up to 32 pins
 - In practice, this number is generally much smaller
- The ports are named A ... E

Teensy 3.5 Schematic

Key take-away: shows us the connection between the Teensy pin numbers and the Arm Cortex M4 I/O ports

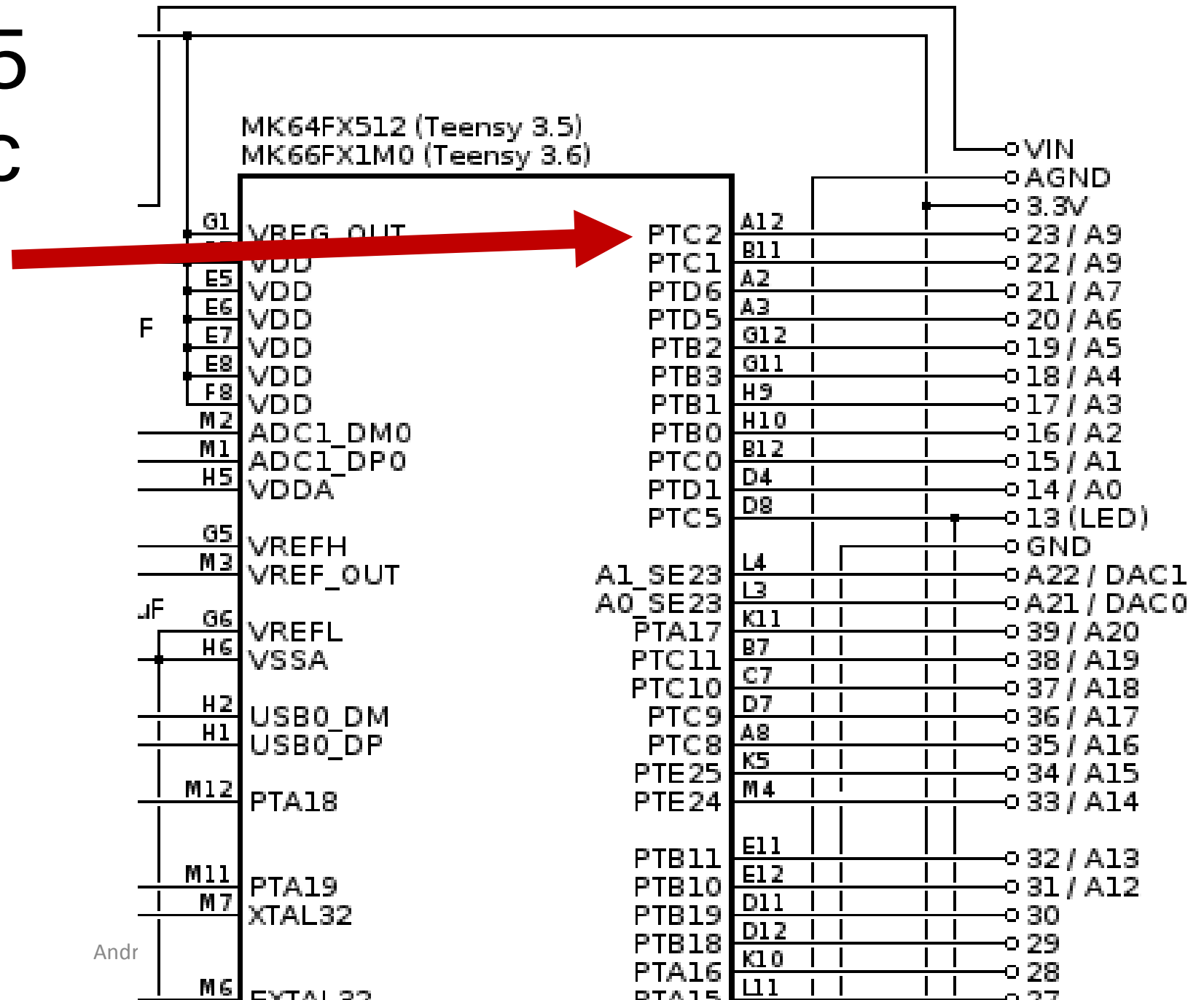


Teensy 3.5 Schematic



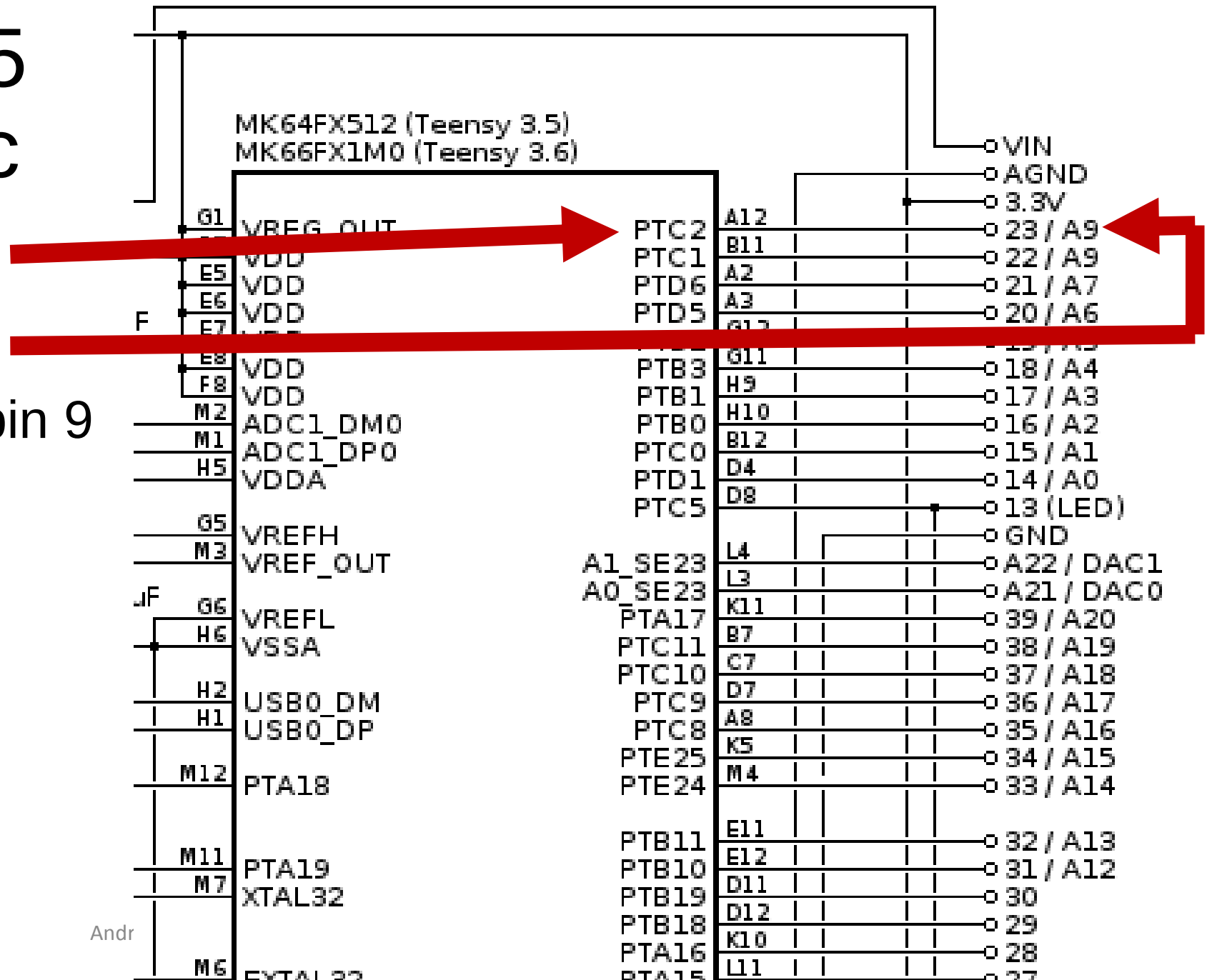
Teensy 3.5 Schematic

- Port C, bit 2



Teensy 3.5 Schematic

- Port C, bit 2
- Teensy pin 23
 - Also analog pin 9



Pins in the Arm Cortex M4

- Most pins have multiple possible functions
 - Can be a digital input or output
 - Some can generate a continuous voltage (analog output)
 - Many can read a continuous voltage (analog input)
 - Communication

... go to Bit Manipulation

Configuring a Pin for Digital Output

There is an on-board LED connected to PORT C, bit 5:
let's write code to blink the LED

Configuring a Pin for Digital Output

There is an on-board LED connected to PORT C, bit 5:
let's write code to blink the LED

- Initialization:

```
// Initialize PORT C, bit 5 to be a digital I/O bit  
PORTC_PCR5 = PORT_PCR_MUX(0x1);
```

- PORTC_PCR5 is a special-purpose register (32 bits) that controls what this specific pin does
- PCR = Port Configuration Register

Configuring a Pin for Digital Output

- Initialization, step 2:

```
// Configure bit 5 to be an output (and leave all others as inputs)
GPIOC_PDDR |= 0x20; // 35
GPIOC_PDDR = GPIOC_PDDR | 0x20;
```

- GPIO = General Purpose Input/Output
- PDDR = Port Data Direction Register
- On boot: all pins are configured as analog inputs

Setting to Pin into the High State

```
// Turn on the bit (and all others unchanged)
GPIOC_PDOR |= 0x20;
```

- The pin is now in a high state
- PDOR = Port Data Output Register

Putting it Together in the Arduino Environment

This function is called when the processor first boots:

```
void setup() {  
    // Configure PORTC, bit 5 to be a digital I/O bit  
    PORTC_PCR5 = PORT_PCR_MUX(0x1);  
  
    // Configure bit 5 to be an output (and leave all others as inputs)  
    GPIOC_PDDR |= 0x20;  
}
```

Putting it Together in the Arduino Environment

And this function is called repeatedly thereafter:

```
void loop() {  
    // Turn on the bit  
    GPIOC_PDOR |= 0x20;  
  
    // Wait for 0.1 second  
    delay(100);  
  
    // Turn off the bit (and all others)  
    GPIOC_PDOR &= ~0x20;  
  
    // Wait for 0.1 second  
    delay(100);  
}
```

Arduino Environment

The environment automatically includes the following function:

```
void main() {  
    setup();  
  
    while(1)  
    {  
        loop();  
    }  
}
```

An Alternative Implementation

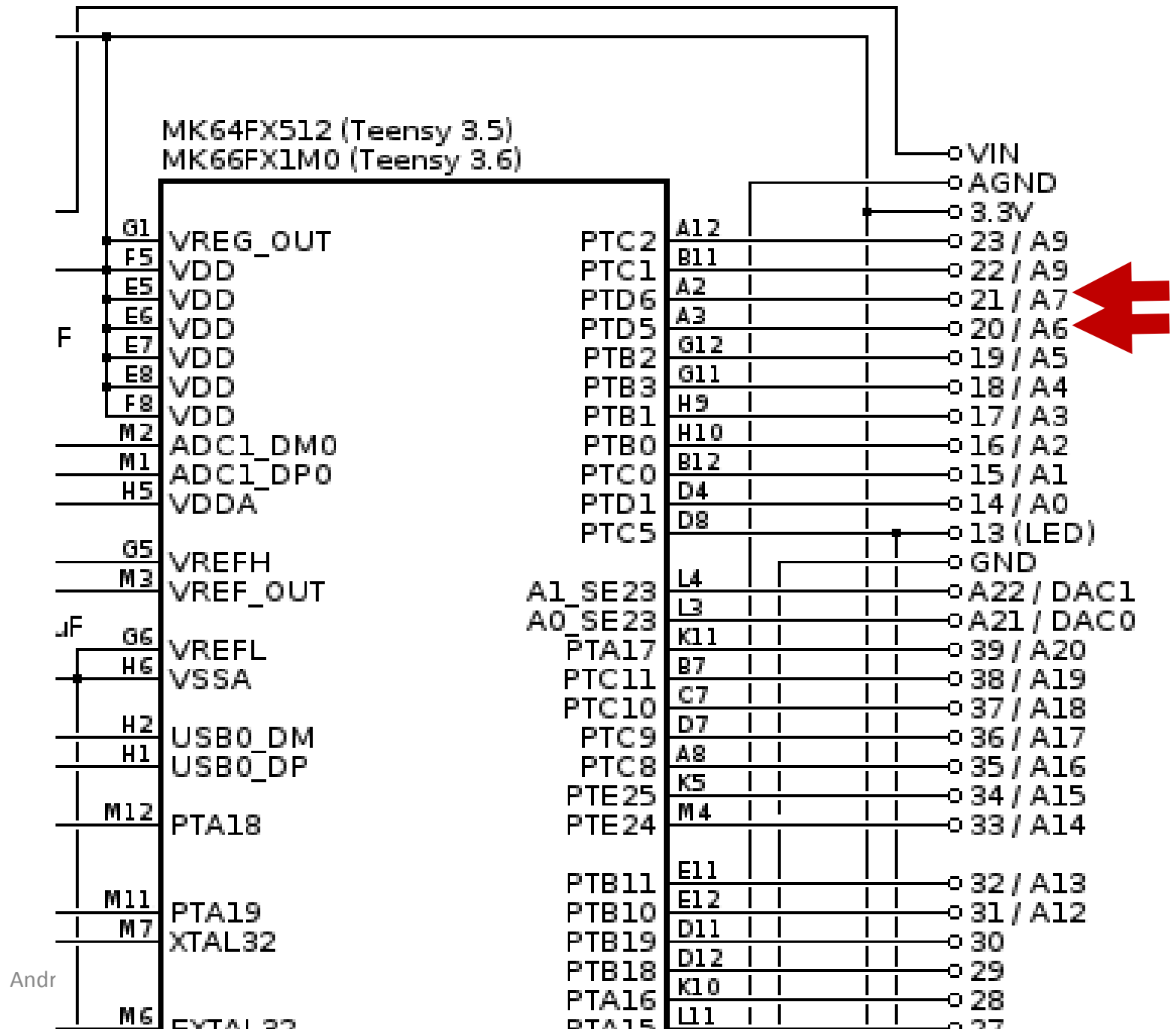
```
void loop() {  
    //  
    GPIOC_PDOR ^= 0x20;  
  
    // Wait for 0.1 second  
    delay(100);  
}
```

PORTS A .. E

- `PORTx_PCRy` = each bit has one register
- `GPIOx_PDDR`, `GPIOx_PDOR`: each port has one register
- Note: the Arduino environment provides other ways to manipulate these pins (don't use these alternatives!)
 - For digital I/O, we will use these registers. We get:
 - Efficiency
 - Simultaneous state change of multiple pins

Teensy 3.5 Schematic

- Let's connect LEDs to PTD5 & 6
- Don't forget the resistor!



Initialization

```
void setup() {  
    // Configure PORTD, pins 5 & 6 as digital I/O  
    PORTD_PCR5 = PORT_PCR_MUX(0x1);  
    PORTD_PCR6 = PORT_PCR_MUX(0x1);  
  
    // Configure bit 5 & 6 to be outputs  
    GPIOD_PDDR |= 0x60;  
}
```

What does this program do?

```
void loop() {  
    GPIOD_PDOR = (GPIOD_PDOR & ~0x60) | 0x60;  
    delay(250);  
    GPIOD_PDOR = (GPIOD_PDOR & ~0x60) | 0x20;  
    delay(250);  
    GPIOD_PDOR = (GPIOD_PDOR & ~0x60) | 0x40;  
    delay(250);  
    GPIOD_PDOR = (GPIOD_PDOR & ~0x60) | 0x0;  
    delay(250);  
}
```


What does this program do?

```
void loop() {  
  GPIOD_PDOR = (GPIOD_PDOR & ~0x60) | 0x60;  
  delay(250);  
  GPIOD_PDOR = (GPIOD_PDOR & ~0x60) | 0x20;  
  delay(250);  
  GPIOD_PDOR = (GPIOD_PDOR & ~0x60) | 0x40;  
  delay(250);  
  GPIOD_PDOR = (GPIOD_PDOR & ~0x60) | 0x0;  
  delay(250);  
}
```

**Flashes LED on PD6 at 2 Hz
on PD5: 1 Hz**

Duty Cycle for each: 50%

Simultaneously Changing the State of Multiple Pins

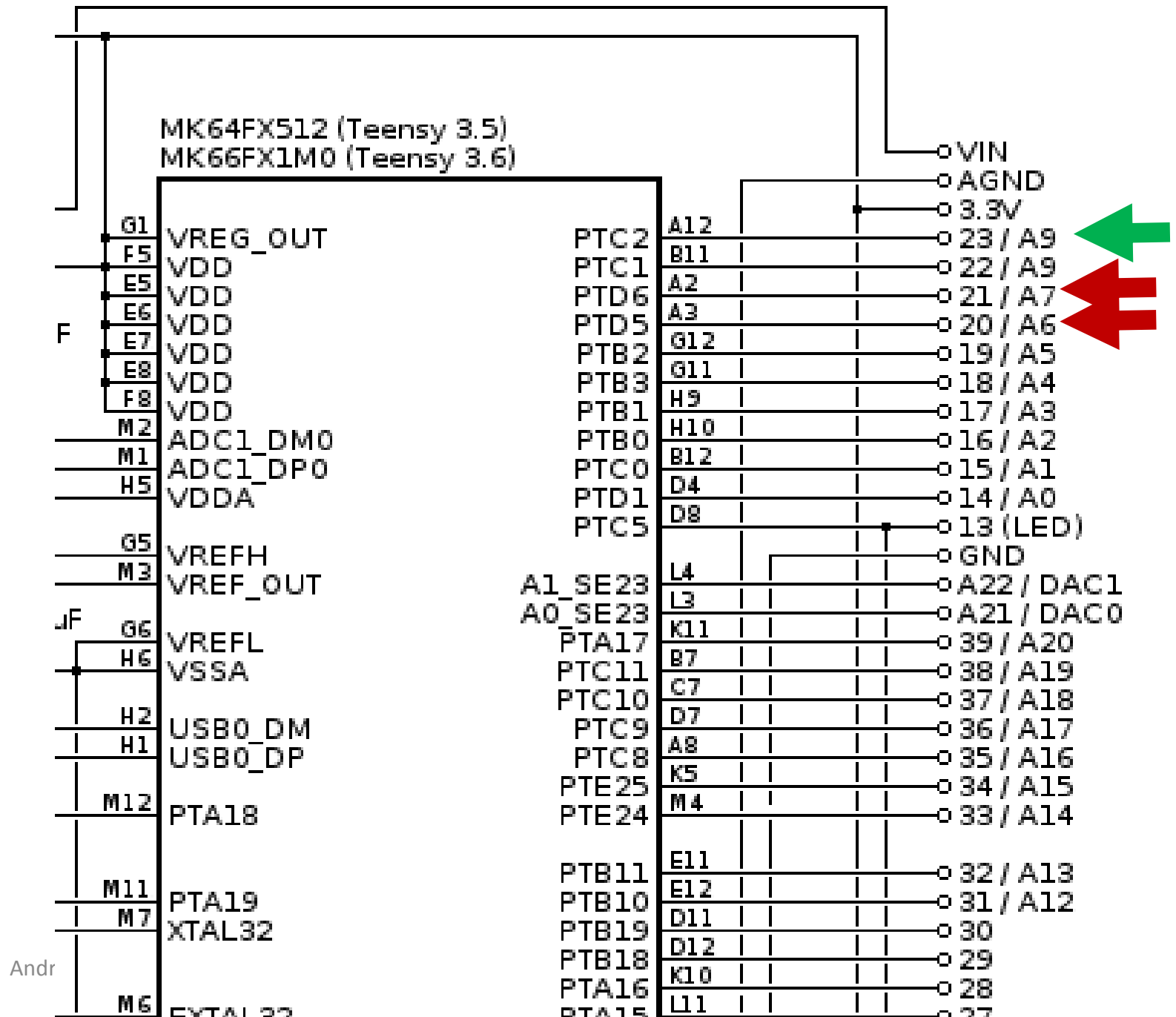
This is the pattern that we want you to follow!

```
GPIOD_PDOR = (GPIOD_PDOR & ~0x60) | 0x20;
```

- “& ~0x60”: set **all** the bits in a group to zeros
- “| 0x20”: set the bits in the group that should be ones
- If you have to make a change later to the set of ones: you just have to change the last number

Teensy 3.5 Schematic

- Let's connect a switch to PTC2
- Don't forget the pull-up resistor!
- If switch reads zero, turn PTD6 on and PTD5 off
- Otherwise, turn PTD6 off and PTD5 on



Initialization

```
void setup() {  
    // Configure PORTD, pins 5 & 6 as digital I/O  
    PORTD_PCR5 = PORT_PCR_MUX(0x1);  
    PORTD_PCR6 = PORT_PCR_MUX(0x1);  
  
    // Configure PORTC, pin 2 as digital I/O  
    PORTC_PCR2 = PORT_PCR_MUX(0x1);  
  
    // Configure bit 5 & 6 to be outputs  
    GPIOD_PDDR |= 0x60;  
}
```

Loop Implementation

```
void loop() {  
    if(GPIOC_PDIR & 0x4)  
    {  
        // Switch open  
        GPIOD_PDOR = ...  
    }else{  
        // Switch closed  
        GPIOD_PDOR = ...  
    }  
}
```

Loop Implementation

```
void loop() {  
    if(GPIOC_PDIR & 0x4)  
    {  
        // Switch open  
        GPIOD_PDOR = (GPIOD_PDOR & ~0x60) | 0x40;  
    }else{  
        // Switch closed  
        GPIOD_PDOR = (GPIOD_PDOR & ~0x60) | 0x20;  
    }  
}
```

