Appendix A: A Teensy Primer

Xiaoguang "Leo" Liu
lxgliu@ucdavis.edu

Last updated: June 25, 2015

1 Introduction

The Teensy is an open-source 32-bit Cortex-M4 based microcontroller development platform in a very small footprint. All the programming and data connection are done through the USB port; a USB cable with a standard "Mini-B" connector is all you need to start using the Teensy!

The Teensy 3.1 features a very capable MK20DX256 32-bit Cortex-M4 processor running at a clock speed of 72 MHz. The MK20DX256 has 256K flash memory, 64K RAM, and 2K EEPROM. The peripherals include 21 analog input, 12 PWM output, 3 UART channels, 2 I2C channels, and 1 SPI channel. Worth noting is that the digital pins are compatible with 5-V input signals, making it easier to interface with older digital components. In addition, the MK20DX256 integrates a 16-bit analog to digital converter (ADC) and a 12-bit digital to analog converter (DAC) that facilitate a wide range of applications where relatively high speed data acquisition and generation are required. For detailed information on the microscontroller, refer to its datasheet [1] and reference manual [2].

Fig. 1 shows a pinout diagram of the platform and Fig. 2 shows the schematic.

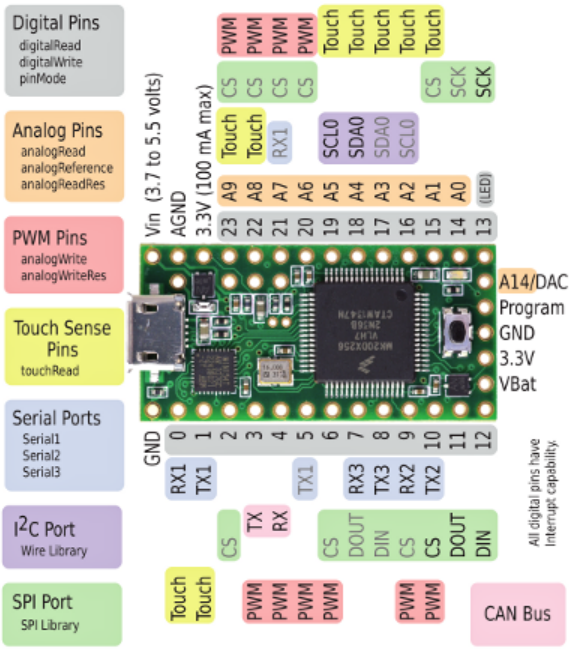

Figure 1: Pinout diagram for the Teensy 3.1 [4].

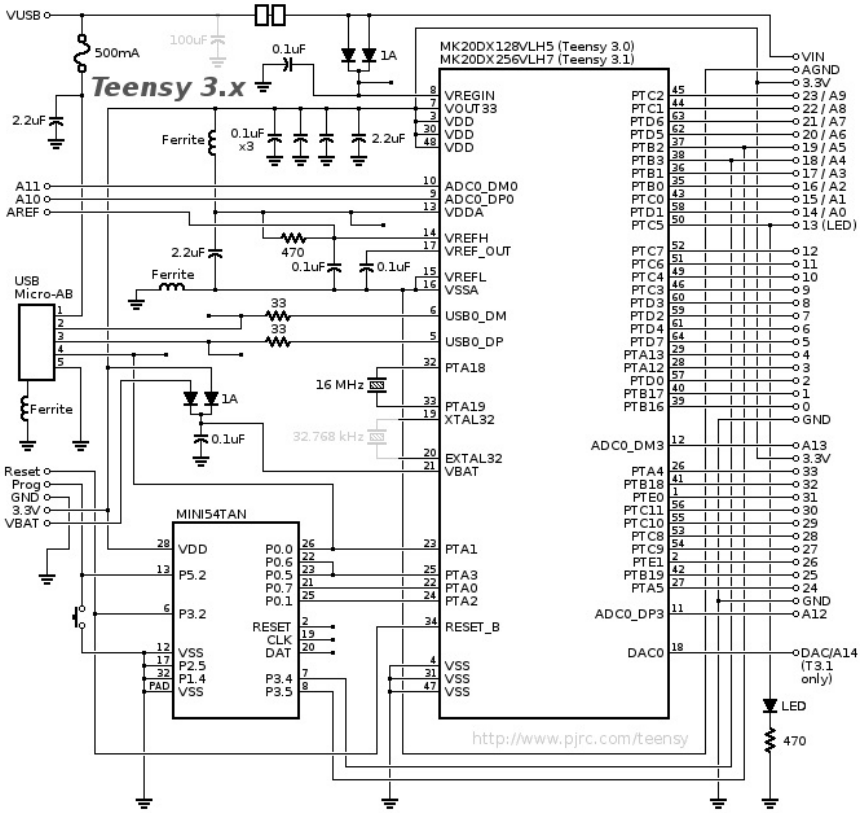

Figure 2: Schematic of the Teensy 3.0/3.1 development platform [3].

The input and output pins of the Teensy are arranged in two parallel rows with a spacing that is compatible with most breadboards. This makes it quite easy to prototype a project where the rest of the components are connected on a breadboard.

The designer of the Teensy platform, Paul J. Stoffregen, is also an avid software developer. Besides designing the hardware of the Teensy board, he also wrote the software libraries to make the Teensy compatible with the Arduino IDEs, which contributes greatly to the popularity of the Teensy platform.

To start using the Teensy 3.1, following the instructions at the PJRC website: `https://www.pjrc.com/teensy/td_download.html`.

2 Microcontroller basics

A microcontrollers is a small computer with all its components, such the processor, memory, and input/output (I/O) interfaces, designed and fabricated on a single integrated circuit (IC). Compared to general purpose computers, a microcontroller usually works at a much lower clock frequency, ranging anywhere between a few kHz to about 200 MHz. The amount of memory available to a microcontroller is also very limited and could be as small as a few kilobyte. Because of the hardware constraints, microcontrollers are usually programmed us A great advantage of using a microcontroller is its very low power consumption. Some microstrollers need only a few mW of power to run at full speed and even less when stand-by mode.

The Teensy Further reading [5, 6].

3 Lighting up an LED

Blinking a Light Emitting Diode (LED) is a canonical example in the embedded system world that shares a same popularity as the "Hello World!" does in computer programming. There are many ways to turn an LED on and off. The simplest among them is to use a digital output pin to apply the necessary voltage across the LED[1].

On the Teensy 3.1, when a pin is set to "digital high", it can source some current and its voltage is roughly equal to VDD (in the range of 1.7 V to 3.6 V). Because of the exponential I-V relationship of the diode, such a large voltage drop across the LED will induce a lot of current, in fact much more than the output circuits in Teensy can handle. Usually a resistor is added to the circuit to prevent this from happening. Similar to a lot of popular microcontroller development platforms, the Teensy has already included an LED with a suitable resistor. If you look at Fig. 2 closely, you will see that an LED and a 470 Ω resistor are connected to pin 13 which is unequivocally labeled as "LED".

So to implement our blinking LED example, we don't actually need to build any circuit. All we need to do is to input the following program[2], compile, and upload to the Teensy. The code is pretty self-explanatory with the added comments.

```
1
  // LED is connected to pin 13
3 int led = 13;

5 // the setup routine initializes the program
  // it runs once when you press reset.
7 void setup() {
  // the pinMode routine set up the function of a pin
9 // it must be called before a pin can be used as either an input or output device
  pinMode(led, OUTPUT);
11 }

13 // the loop routine runs over and over again forever:
  void loop() {
15 // the digitalWrite routine writes a logic HIGH or LOW value to a digital pin
  // On the Teensy 3.1, a logic HIGH output has a minimum value of (VDD-0.5) V and a
     logic LOW has a maximum value of 0.5 V.
17 digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
19 digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
21 }
```

Here is the Teensy board in action. Unfortunately you can't really see the LED blink.

[1] An LED is basically a diode that can emit light when forward biased (anode at a higher electric potential than cathode). The colors of the emitted light of an LED depends on the material composition and the structure of the diode design. The forward voltage drop will also be different for LEDs with different colors.

[2] The program can be accessed through the Arduino IDE by going to "File->Examples->Basics->Blink"

4 Reading digital input

5 Using PWM for analog output

6 Using interrupts

Another reason for using interrupts is to ensure the response time for certain tasks. Microcontrollers are used a lot in industrial control systems, where certain sensor inputs must be attended to within a predescribed amount of time to ensure the stability of the control algorithm. Through dedicated hardware, an interrupt triggers the microcontroller to halt the less important task in order to take care of high priority tasks in a timely fashion.

7 Using the timer

8 Using the digital to analog converter (DAC)

9 Using the memory

10 Using the analog to digital converter (ADC)

11 Using the audio library

References

[1] Freescale Semiconductor, Inc., "K20 Sub-Family Data Sheet", available: `https://www.pjrc.com/teensy/K20P64M72SF1.pdf`, accessed Jun. 24, 2015.

[2] Freescale Semiconductor, Inc., "K20 Sub-Family Reference Manual", available: `https://www.pjrc.com/teensy/K20P64M72SF1RM.pdf`, accessed Jun. 25, 2015.

[3] PJRC.com, LLC, "Teensy reference: schematic", online: `https://www.pjrc.com/teensy/schematic.html`, accessed Jun. 23, 2015.

[4] PJRC.com, LLC, "Teensy: pinouts", online: `https://www.pjrc.com/teensy/teensy31.html`, accessed Jun. 23, 2015.

[5] Milan Verle, "Chapter 1: Introduction to Microcontrollers," *Architecture and programming of 8051 MCU's*, online: `http://www.mikroe.com/chapters/view/64/chapter-1-introduction-to-microcontrollers/`, accessed Jun. 24, 2015.

[6] Gunther Gridling and Bettina Weiss, "Introduction to Microcontrollers," *Courses 182.064 & 182.074*, Institute of Computer Engineering, Vienna University of Technology, Feb. 26, 2007.