Modern Assembly Language Programming with the ARM processor

Chapter 12: Pulse Modulation

1. Introduction
2. Raspberry Pi PWM
3. pcDuino PWM

Pulse Modulation

The GPIO device can send a digital signal, but what if you need to send an analog signal?

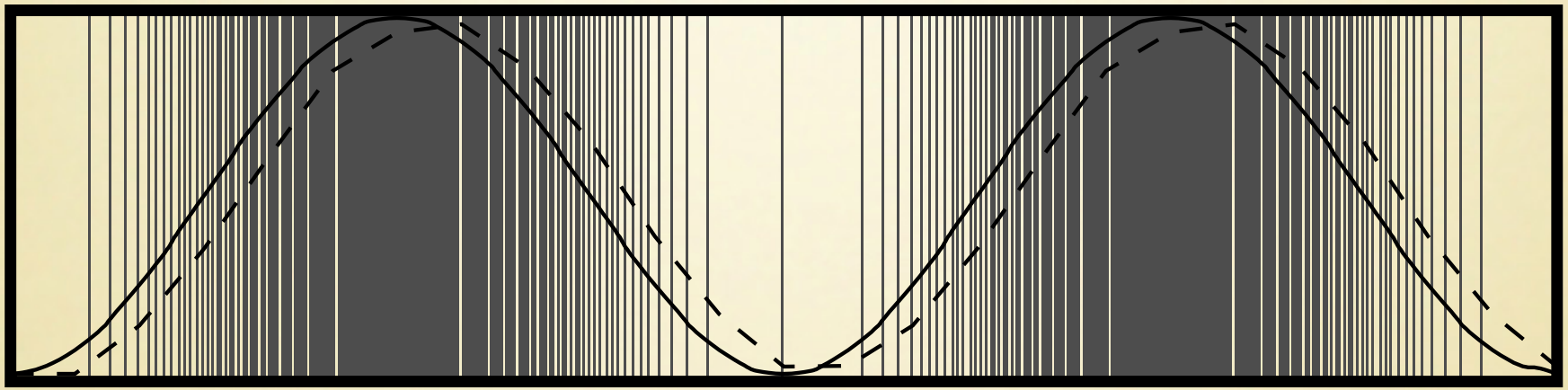
- Send a stream of pulses to the device.
- The device acts as a low-pass filter, which averages the digital pulses into an analog voltage.
- By varying the percentage of time that the pulses are high versus low, the computer can control how much average energy is sent to the device.
- The percentage of time that the pulses are high versus low is known as the *duty cycle*.
- Varying the duty cycle is referred to as *modulation*.

Types of Pulse Modulation

There are two major types of pulse modulation.

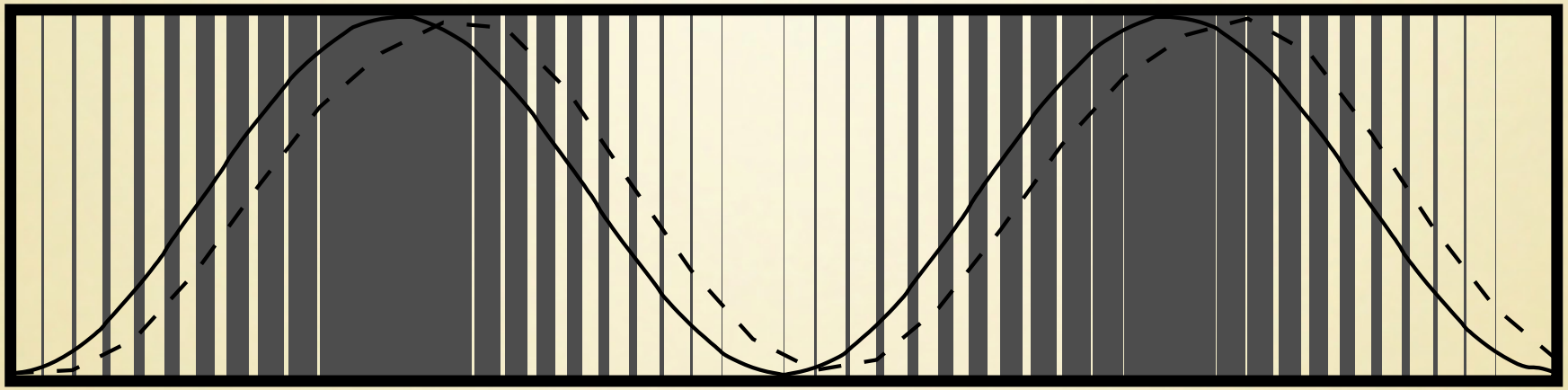
- With Pulse Density Modulation (PDM), the pulses of energy are of constant duration, and the time between them is modulated.
- With Pulse Width Modulation (PWM), time between pulses is constant and the length of time that the signal is high is modulated.

Pulse Frequency Modulation



Pulse: Modulation Signal: Received Signal:

Pulse Width Modulation



Pulse: ▬▬ Modulation Signal: ——— Received Signal: – – –

Programming Pulse Modulation

Most pulse modulation devices are configured in three steps as follows:

- The base frequency of the clock that drives the PWM device is configured. This step is usually optional.
- The mode of operation for the pulse modulation device is configured by writing to one or more configuration registers in the pulse modulation device.
- The cycle time is set by writing a “range” value into a register in the pulse modulation device. This value is usually set as a multiple of the base clock cycle time.

Once the device is configured, the duty cycle can be changed easily by writing to one or more registers in the pulse modulation device.

Raspberry Pi PWM Registers

Offset	Name	Description	Size	R/W
00_{16}	PWMCTL	PWM Control	32	R/W
04_{16}	PWMSTA	PWM FIFO Status	32	R/W
08_{16}	PWMDMAC	PWM DMA Configuration	32	R/W
10_{16}	PWMRNG1	PWM Channel 1 Range	32	R/W
14_{16}	PWMDAT1	PWM Channel 1 Data	32	R/W
18_{16}	PWMFIF1	PWM FIFO Input	32	R/W
20_{16}	PWMRNG2	PWM Channel 2 Range	32	R/W
24_{16}	PWMDAT2	PWM Channel 2 Data	32	R/W

Programming the Raspberry Pi PWM Device

There are three modes of operation for the BCM2835 PWM device:

- pulse density modulation mode,
- pulse width modulation mode, and
- serial transmission mode.

Control Register - PWM1

Bit	Name	Description	Values
0	PWEN1	Channel 1 Enable	0: Channel is disabled 1: Channel is enabled
1	MODE1	Channel 1 Mode	0: PDM or PWM mode 1: Serial mode
2	RPTL1	Channel 1 Repeat Last	0: Transmission stops when FIFO empty 1: Last data is sent repeatedly
3	SBIT1	Channel 1 Silence Bit	0: Output goes low when not transmitting 1: Output goes high when not transmitting
4	POLA1	Channel 1 Polarity	0: 0 is low voltage and 1 is high voltage 1: 1 is low voltage and 0 is high voltage
5	USEF1	Channel 1 Use FIFO	0: Data register is used 1: FIFO is used
6	CLRF1	Channel 1 Clear FIFO	0: No effect 1: Causes FIFO to be emptied
7	MSEN1	Channel 1 PWM Enable	0: PDM mode 1: PWM mode

Control Register - PWM2

Bit	Name	Description	Values
8	PWEN2	Channel 2 Enable	0: Channel is disabled 1: Channel is enabled
9	MODE2	Channel 2 Mode	0: PDM or PWM mode 1: Serial mode
10	RPTL2	Channel 2 Repeat Last	0: Transmission stops when FIFO empty 1: Last data is sent repeatedly
11	SBIT2	Channel 2 Silence Bit	0: Output goes low when not transmitting 1: Output goes high when not transmitting
12	POLA2	Channel 2 Polarity	0: 0 is low voltage and 1 is high voltage 1: 1 is low voltage and 0 is high voltage
13	USEF2	Channel 2 Use FIFO	0: Data register is used 1: FIFO is used
14	Unused	Reserved	
15	MSEN2	Channel 2 PWM Enable	0: PDM mode 1: PWM mode
16-31	Unused	Reserved	

Initialization on the Raspberry Pi

For a base frequency of 100 KHz, the steps would be as follows:

1. Verify that the clock manager device is configured to send a 100 MHz clock to the pulse modulator device through PWM_CLK.
2. Store 1000 in the PWMRNG1 register to divide PWM_CLK by 1000.
3. Initialize the duty cycle to 0%by writing zero to the PWMDAT1 register.
4. Enable PWM channel 1 to operate in PWM mode by writing to PWMCTL:
   - set bit zero to 1,
   - bit one to 0,
   - set bit five to 0, and
   - set bit seven of to 1.

Once this initialization is performed, we can set or change the duty cycle at any time by writing a value between 0 and 1000 to the PWMDAT1 register.

pcDuino PWM register map.

Offset	Name	Description
200_{16}	PWMCTL	PWM Control
204_{16}	PWM_CH0_PERIOD	PWM Channel 0 Period
208_{16}	PWM_CH1_PERIOD	PWM Channel 1 Period

pcDuino PWM Control Register - Channel 0

Bit	Name	Description	Values
3-0	CH0_PRESCAL	Channel 0 Prescale	These bits must be set before the clock is enabled.
4	CH0_EN	Channel 0 Enable	0: Channel disabled 1: Channel enabled
5	CH0_ACT_STA	Channel 0 Polarity	0: Channel is active low 1: Channel is active high
6	SCLK_CH0_GATING	Channel 0 Clock	0: Clock disabled 1: Clock enabled
7	CH0_PUL_START	Start pulse	If the PWM device is configured for pulse mode, writing a 1 to this bit causes the PWM device to emit a single pulse.
8	PWM0_BYPASS	Bypass PWM	0: Output PWM device signal 1: Output base clock
9	SCLK_CH0_MODE	Select Mode	0: PWM mode 1: Pulse mode
10-14		Not Used	

pcDuino PWM Control Register - Channel 1

Bit	Name	Description	Values
18-15	CH1_PRESCAL	Channel 1 Prescale	These bits must be set before the clock is enabled.
19	CH1_EN	Channel 1 Enable	0: Channel disabled 1: Channel enabled
20	CH1_ACT_STA	Channel 1 Polarity	0: Channel is active low 1: Channel is active high
21	SCLK_CH1_GATING	Channel 1 Clock	0: Clock disabled 1: Clock enabled
22	CH1_PUL_START	Start pulse	If the PWM device is configured for pulse mode, writing a 1 to this bit causes the PWM device to emit a single pulse.
23	PWM1_BYPASS	Bypass PWM	0: Output PWM device signal 1: Output base clock
24	SCLK_CH1_MODE	Select Mode	0: PWM mode 1: Pulse mode
31-25		Not Used	

Period and Duty Cycle

The two period registers are each organized as two 16-bit numbers:

- The upper 16 bits control the total number of clock cycles in one period.
- The lower 16 bits of the channel period register control the duty cycle.

Setting Base Frequency

The PWM frequency is calculated as

$$f = \frac{\frac{OSC24M}{PSC}}{N+1},$$

where $OSC24M$ is the frequency of the base clock (the default is 24MHz), PSC is the prescale value set in the channel prescale bits in the PWM control register, and N is the value stored in the upper 16 bits of the channel period register.

Setting Duty Cycle

The lower 16 bits of the channel period register control the duty cycle. The duty cycle (expressed as % of full on) can be calculated as

$$d = \frac{D}{N} \times 100,$$

where D is the value stored in the lower 16 bits of the channel period register. Note: the condition $D \leq N$ must always remain true. If the programmer allows D to become greater than N, the results are unpredictable.

Configuring pcDunio PWM

1. Disable the desired channel:
   1. Read the PWM control register into x.
   2. Clear all of the bits in x for the desired PWM channel.
   3. Write x back to the PWM control register
2. Initialize the period register for the desired channel.
   1. Calculate the desired value for N.
   2. Let $D = 0$.
   3. Let $y = N \times 2^{16} + D$
   4. Write y to the desired channel period register.
3. Set the prescaler.
   1. Select the four-bit code for the desired divisor from the table in the textbook.
   2. Set the prescaler code bits in x.
   3. Write x back to the PWM control register
4. Enable the PWM device.
   1. Set the appropriate bits in x to enable the desired channel, select the polarity, and enable the clock.
   2. Write x to the PWM control register.

Chapter Summary

Pulse modulation is

- a group of methods for generating analog signals using digital equipment.
- commonly used in control systems to regulate the power sent to motors and other devices.

The cycle frequency must be programmed to match the application.

It can take some experimentation to find the best frequency for any given application.