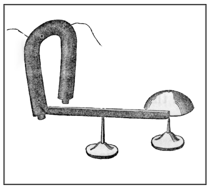
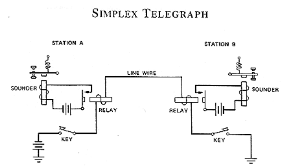
Joseph Henry & the Telegraph





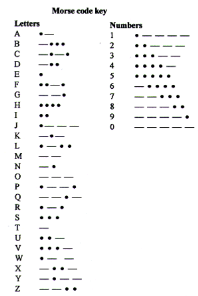
Elementary neutral telegraph circuit.

- Albany Academy Experiment
- Assisted Morse at Princeton
- 1st Head of Smithsonian
- Unit of inductance: Henry

CSE466 1

Morse Code

- Simple sequences of short and long clicks to represent letters and numbers
- Easier to generate than sound
- Easier to distinguish than sound



CSE466 2

UPC Codes

•Bars come in four widths 1-2-3-4

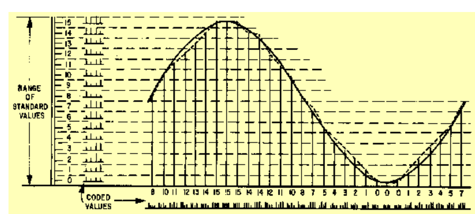
•Start is 1-1-1 (black-white-black)

- 0 = 3-2-1-1
- 1 = 2-2-2-1
- 2 = 2-1-2-2
- 3 = 1-4-1-1
- 4 = 1-1-3-2
- 5 = 1-2-3-1
- 6 = 1-1-1-4
- 7 = 1-3-1-2
- 8 = 1-2-1-3
- 9 = 3-1-1-2

- The zero is 3-2-1-1 (space-bar-space-bar).
- The four is 1-1-3-2 (space-bar-space-bar).
- The three is 1-4-1-1 (space-bar-space-bar).
- The next three zeros are 3-2-1-1 (space-bar-space-bar).
- In the middle there is a standard 1-1-1-1-1 (space-bar-space-bar-space), which is important because it means the numbers on the right are optically inverted!
- The one is 2-2-2-1 (bar-space-bar-space).
- …
- The stop character is a 1-1-1 (bar-space-bar).

CSE466 3

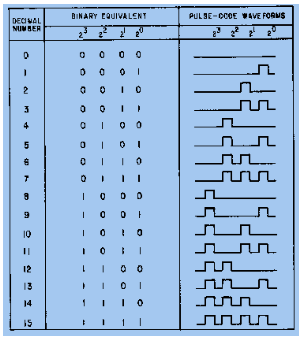
Communicating With Pulses



- PCM: Pulse Code Modulation

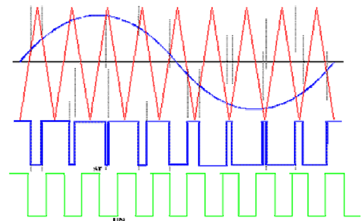
CSE466 4

PCM: Pulse Code Modulation



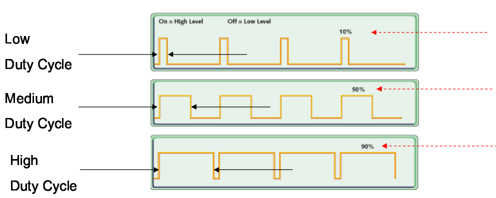
DECIMAL NUMBER	BINARY EQUIVALENT 2^3	2^2	2^1	2^0	PULSE-CODE WAVEFORMS
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	

PWM: Pulse Width Modulation



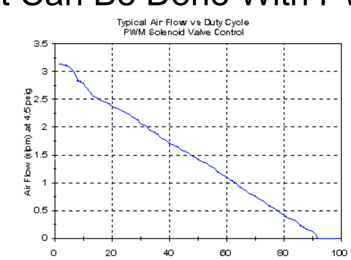
- Signal is compared to a sawtooth wave producing a pulse width proportional to amplitude

What Can Be Done With PWM?



- Question: What happens if voltages like the ones above are connected to a light bulb?
- Answer: The longer the duty cycle, the longer the light bulb is on and the brighter the light.

What Can Be Done With PWM?



- Average power can be controlled
- Average flows can also be controlled by fully opening and closing a valve with some duty cycle

PULSE WIDTH MODULATION

- Pulse Width Modulation (PWM) involves the generation of a series of pulses at a fixed period and frequency.
- The duty cycle defines the width of each pulse which is varied to generate waveforms.

PULSE WIDTH MODULATION

- A simple low pass filter is then used to generate an output voltage directly pro-portional to the average time spent in the HIGH state.
- (i.e., 50% duty cycle is equal to 2.5 volts when VDD =5.0V).

RC low pass filter (integrator)

Choosing the -3 dB point at 4 kHz, and using the Relation:

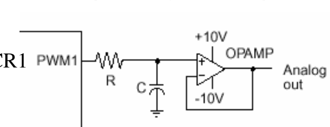
RC = 1/(2 · pi ·f)

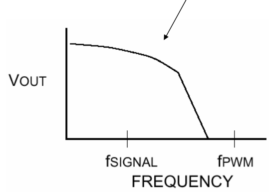
we get R = 4 k, if C is chosen as 0.01 mF:

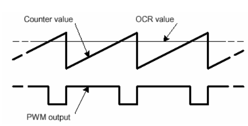
R = 4.0 k

C = 0.01 mF

OCR1

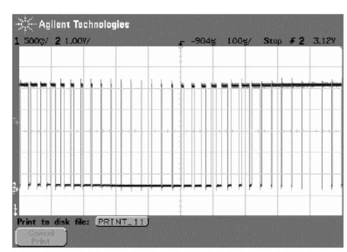




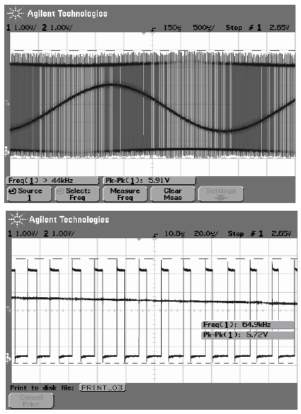


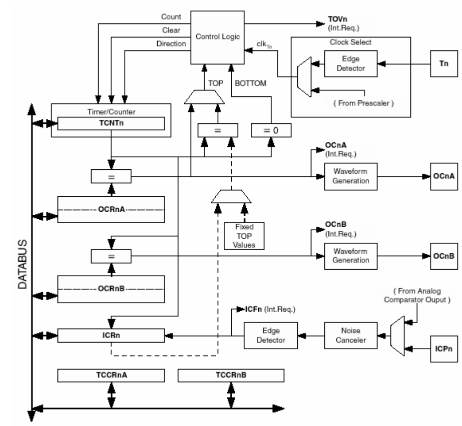
AVR PWM

- AVR counter/timer 1
- To change the PWM base frequency, the timer clock frequency and top counter value is changed.
- Faster clock and/or lower top value will increase the PWM base frequency, or timer overflow frequency.
- With full resolution (top value 255) the maximum PWM base frequency is 250 kHz.
- Altering the value of the Output Compare Registers (OCR) changes the duty cycle.
- Increasing the OCR value increases the duty cycle. The PWM output is high until the OCR value is reached, and low until the timer reaches the top value and wraps back to 0



Filtered and unfiltered PWM Outputs





PWM Timer 1 Example

```
/* Controls a LED that can be directly connected from OC1A
to GND.  The brightness of the LED is controlled with the
PWM.  After each period of the PWM, the PWM value is either
incremented or decremented */
int
main (void)
{
   ioinit ();
   /* loop forever, the interrupts are doing the rest */
   for (;;)
      ;
   return (0);
}
```



```
void
ioinit (void)
{
   /* tmr1 is 10-bit PWM */
   TCCR1A = _BV (PWM10) | _BV (PWM11) | _BV (COM1A1);
   /* tmr1 running on full MCU clock */
   TCCR1B = _BV (CS10);
   /* set PWM value to 0 */
   OCR1A = 0;
   /* enable OC1 and PB2 as output */
   DDRD = _BV (PD5);
   timer_enable_int (_BV (TOIE1));
   /* enable interrupts */
   sei ();
}
```

```
SIGNAL (SIG_OVERFLOW1)
{
   switch (direction)
   {
      case UP:
         if (++pwm == 1023)
            direction = DOWN;
         break;

      case DOWN:
         if (--pwm == 0)
            direction = UP;
         break;
   }
   OCR1A = pwm; /* triangle */
}
```


R-2R DAC