Micro controller

1. micro processor Vs Micro - controller

ⅰ, micro processor : .cpu for computer

- processing (computing, data management)
- No RAM, ROM, I/O on CPU chip itself
- ex, Intel, AMD, ARM

ⅱ, Micro - controller : integrates a number of components of a microprocessor onto a single micro chip

: CPU core, Memory (ROM & RAM), I/O, peripherals

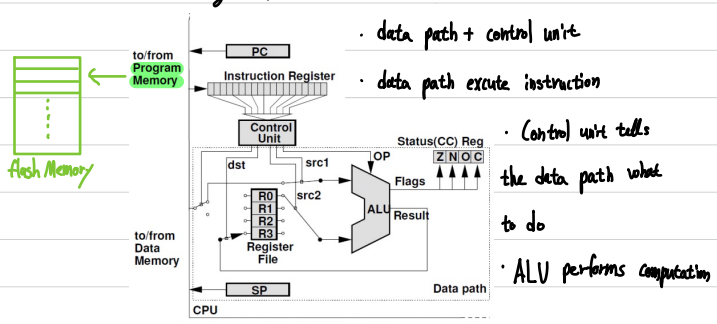
	Microprocessor 높은 계산	Microcontroller
Applications	General computing (i.e. Laptops, Desktops)	Appliances, Embedded systems
Speed	Very fast	Relatively slow 16MHz로
External parts	Many	Few
Cost	High	Low
Energy Use	Medium to high	Very low to low 배터리로 가능
Vendors	Intel, AMD, ARM	ATMEL, ST, Texas Instruments, Microchips

2. Peripherals

- GPIO (general purpose input and output)
- Timer : to perform time-specific task
- Communication peripherals : UART, SPI, I2C, USB

USART : Universal synchronous and Asynchronous serial Receiver and Transmitter

3. CPU (Central Processing Unit)

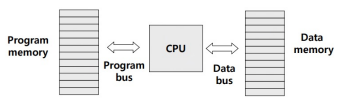


Basic CPU architecture

- data path + control unit
- data path excute instruction
- (control unit tells the data path what to do
- ALU performs computation

4. Harvard Architecture

- Seperate program bus and data bus
- Instruction pipelining easy



5. RISC : Reduced Instruction set computer

1. Relatively few instructions 꽤 필요한 적은 명령
2. Relatively few addressing modes
3. Memory access limited to load and store instructions
4. All operations done within the registers of the CPU
5. Fixed-length, easily decoded instruction format
6. Single-cycle instruction execution pipline
7. Hardwired rather than microprogrammed control 속도↑
8. A relatively large number of registers in the processor unit
9. Efficient instruction pipeline
10. Compiler support for efficient translation of high-level language programs into machine language programs.

The concept of RISC architecture :
to reduce execution time by simplifying the instruction set of the computer

6. AVR Micro - controller family

ⅰ, AVR Architecture

- Harvard architecture, 8-bit Risc, single chip
- Few instructions : 130
- only a few addressing modes (주소 지정 방식)
- execute 1 instruction in 1 internal clock cycle

ⅱ, ATmega2560

1, AVR 8-bit microcontroller. CPU와 SRAM 사이에 연결된 데이터 버스 크기, 즉 CPU에서 SRAM을 한번에 얼마나 큰 데이터를 읽을 수 있는지를 의미

2, Advanced RISC architecture

- 135 Instructions - most single clock cycle execution.
- 32X8 general purpose registers
- 16MHz

3, Program and data memory

- 256kB program memory (Flash), 8kB SRAM
- 8kB SRAM (기억장치)

4, Peripheral features

- two 8-bit timer/counter, four 16-bit timer/counter
- four 8-bit PWM channels, twelve 2~16 bits PWM channels
- 16 10-bit ADC channels
- four programmable serial USART

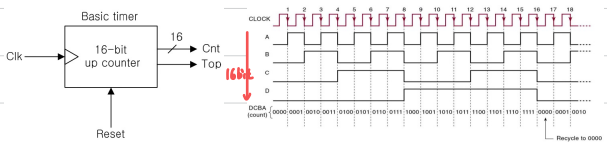
임베디드 시스템 설계

7-1 Timer/counter

1. Timer

ⅰ, concept : measures time intervals

- to generate timed output events
- to measure input events

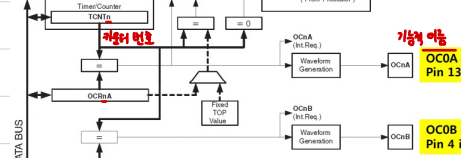


ⅱ, clock pulses

- Clk period : 10 ns (resolution), count 40,000 clk pulses ⇒ $2\times10^{4} \times 10\times10^{-9} = 200\times10^{-6}$ (200 micro seconds)
- 16-bit counter : 65,535 x 10ns (0, 1, ⋯, $2^{16}-1 = 65535$)
- Top : indicates top count reached, wrap-around

2. Counter

ⅰ, concept : counts pulses on a general input signal rather than periodic clock pulse

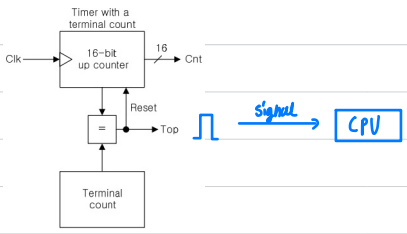


3. Interval timer

ⅰ, concept : Indicates when desired time interval has passed (alarm)

ⅱ, we set terminal count to desired interval

Number of clock cycles = Desired time interval / Clock period



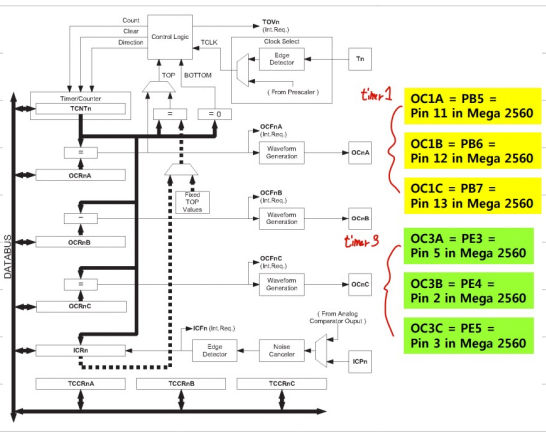
4. Cascaded counters (counter 레지스터 2개, 16bit - 16bit ⇒ 32bit)

ⅰ, concept : Prescaler
- Divides clock
- Increases range, decreases resolution

5. Block diagram

ⅰ, 8-bit (0,2)

ⅱ, 16 bit (1,3,4,5)



cf, The BV macro

Using macros make this easier to read (BV= Bit value)

```
#define _BV(x)  (1<<x)
```

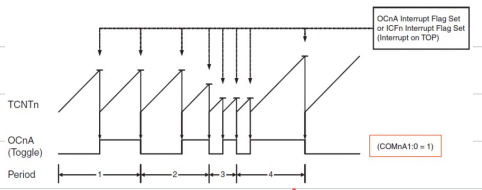
```
PORTC |= _BV(0);      // set bit 0 only
PORTC &= ~(_BV(1)); // clear bit 0 only
PORTC |= _BV(7);      // Set bit 7 only
```

6. Timer/Counter Modes

ⅰ, Normal mode : counting direction is always up, no counter clear is performed

ⅱ, CTC Mode : Clear timer on compare match

- the counter is cleared to zero when the counter value (TCNTn) matches either OCRnA or ICRn
- the OCRnA or ICRn define the top value for counter, hence also its resolution



$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$ ($\frac{16\times10^{6}}{N}$: 마이크로 컨트롤러 주파수, N : prescale 값)

0 1 2 ⇒ 3개

0 1 2 ⋯ OCR ⇒ OCR+1

7. Related Registers

7	6	5	4	3	2	1	0	
COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B

Bits	Functions
WGM13,WGM12,WGM11,WGM10	Timer/Counter mode selection
COM1A1,COM1A0	Compare output mode selection
CS12,CS11,CS10	Prescaler

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation	TOP	Update of OCRnx at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	TOP	TOP
15	1	1	1	1	Fast PWM	OCRnA	TOP	TOP

- Compare output mode selection

COMnA1/COMnB1/ COMnC1	COMnA0/COMnB0/ COMnC0	Description
0	0	Normal port operation, OCnA/OCnB/OCnC disconnected.
0	1	Toggle OCnA/OCnB/OCnC on compare match.
1	0	Clear OCnA/OCnB/OCnC on compare match (set output to low level).
1	1	Set OCnA/OCnB/OCnC on compare match (set output to high level).

- Clock Select bit description

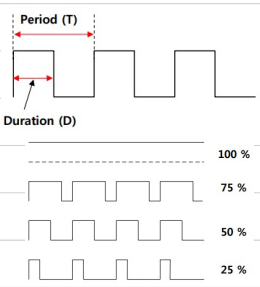
CSn2	CSn1	CSn0	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

7-2 PWM

1. PWM (Pulse Width Modulation)

ⅰ, concept : information is represented by the width of pulse

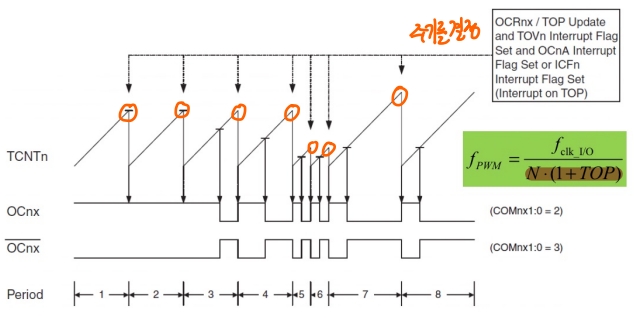
- Duty ratio = Duration/Period
- PWM frequency = 1/T
- When connected to LPF, it acts as a DAC



2. Mode

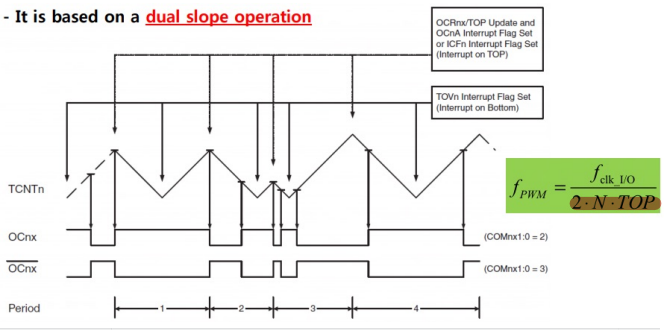
ⅰ, Fast PWM mode

- It provides a high frequency PWM
- Singl - slope operation



ⅱ, Phase Correct PWM mode

- It is based on a dual slope operation
- dual - slope operation



8. Interrupt

1, Concept

- events that require immediate attention by the microcontroller
- MC pause its current task and attend to interrupt by executing an Interrupt Service Routine (ISR)
- At the end of the ISR, MC returns to the task it had pause and continue its normal operations

- In order for the microcontroller to respond to an interrupt event the interrupt feature of the microcontroller must be enabled along with the specific interrupt. This is done by setting the Global Interrupt Enabled bit and the Interrupt Enable bit of the specific interrupt.

- *Interrupt Flags and Enabled bits*

  Each interrupt is associated with two bits, an Interrupt Flag Bit and an Interrupt Enabled Bit. These bits are located in the I/O registers associated with the specific interrupt:

  - The interrupt flag bit is set whenever the interrupt event occur, whether or not the interrupt is enabled

  - The interrupt enabled bit is used to enable or disable a specific interrupt. Basically is tells the microcontroller whether or not it should respond to the interrupt if it is triggered.

Both the Interrupt Flag and the Interrupt Enabled are required for an interrupt request to be generated

- *Global Interrupt Enabled Bit*

  Apart from the enabled bits for the specific interrupts the global interrupt enabled bit MUST be enabled for interrupts to be activated in the microcontoller. For the AVR 8-bits microcontroller this bit is located in the Status I/O Register (SREG). The Global Interrupt Enabled is bit 7, the I bit, in the SREG.

  - sei() : Sets the global interrupt enable bit.
  - cli() : Clears the global interrupt enable bit.

SREG

7	6	5	4	3	2	1	0
I	T	H	S	V	N	Z	C

2. Interrupt Programming

- In AVR-GCC environment, the vector table is predefined to point to interrupt routines with predetermined names
- for ATMega 2560

Number	vector name	Interrupt request condition
0	RESET_vect	External Pin, Power-on reset,...
1	INT0_vect	Externa Interrupt Request 0
2	INT1_vect	Externa Interrupt Request 1
3	INT2_vect	Externa Interrupt Request 2
4	INT3_vect	Externa Interrupt Request 3
5	INT4_vect	Externa Interrupt Request 4
6	INT5_vect	Externa Interrupt Request 5
7	INT6_vect	Externa Interrupt Request 6
8	INT7_vect	Externa Interrupt Request 7
9	PCINT0_vect	Pin Change Interrupt Request 0
10	PCINT1_vect	Pin Change Interrupt Request 1
11	PCINT2_vect	Pin Change Interrupt Request 2
12	WDT_vect	Watchdog Time-out Interrupt
13	TIMER2_COMPA_vect	Timer/Counter2 Compare Match A
14	TIMER2_COMPB_vect	Timer/Counter2 Compare Match B
15	TIMER2_OVF_vect	Timer/Counter2 Overflow
16	TIMER1_CAPT_vect	Timer/Counter1 Captuer Event
17	TIMER1_COMPA_vect	Timer/Counter1 Compare Match A

3. Timer Interrupt

- Interrupts requested by timer/counters.
- Timer interrupts allow us to perform a task at very specifically timed intervals regardless of what else is going on in our code
- For setting up a timer interrupt, we use a timer/counter in CTC mode.

- What to do
  - Configure Timer/Counter1 as CTC mode (TCCR1A)
  - Set the prescaler value (TCCR1B)
  - Set the top value (OCR1A)
  - Clear Timer/Counter1 (TCNT1)
  - Enable output compare match interrupt (TIMSK1)
  - Clear output compare A match flag (TIFR1)
  - Enable global interrupt : sei()

4. External Interrupt

I, Problem

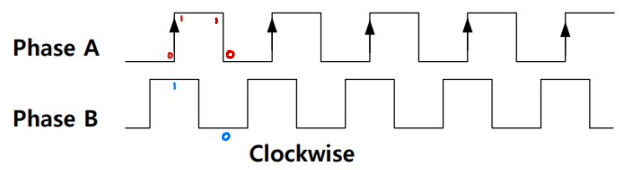
Make an LCD-based counter that counts the click number of a rotary encoder.
Clockwise rotation → counter value increases
Counter-clockwise rotation → counter value decreases.

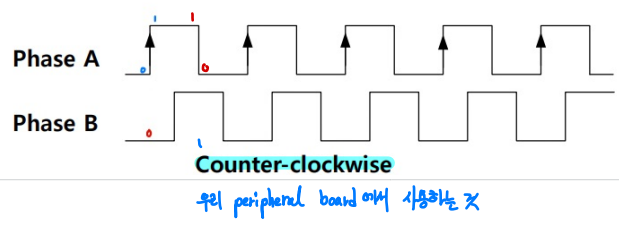
II, Rotary Encoder

- A sensor that is used to measure the angular displacement of a rotational axis.
- Sometimes, it is used to construct a user interface.

- principles
  - When it is rotated, it generates two signals, which are called Phase A and Phase B.
  - The two signals are out of phase as shown below.
  - We can decide the rotational direction by checking out the wave patterns.

한 바퀴 돌며 edge가 몇개 생길 지는 pre-defined 되어 있다.
⇒ 발생된 edge 수로 회전 방향을 알아냄





우리 peripheral board에서 사용하는 것

III, What to do

인터럽트 pin 결정

- Configure the external interrupt sense control (EICRA)
- Clear external interrupt flag (EIFR)
- Enable the external interrupt (EIMSK)
- Define the external interrupt service routine

```
ISR(INTn_vect)  // n=0,1,2,3
{
   //Interrupt service routine goes here.
}
```

- Enable global interrupt : sei()

EICRA : External interrupt control register A
EICRB : External interrupt control register B

Bit	7	6	5	4	3	2	1	0	
(0x69)	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
(0x6A)	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	EICRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Interrupt Sense Control(1)

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request.
0	1	Any edge of INTn generates asynchronously an interrupt request.
1	0	The falling edge of INTn generates asynchronously an interrupt request.
1	1	The rising edge of INTn generates asynchronously an interrupt request.

EIFR : External interrupt flag register

interrupt 신호 방향성 0 → 1

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	IINTF0	EIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

the flag can be cleared by writing a logical one to it

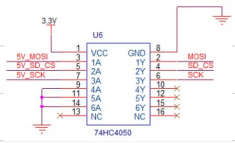
EIMSK : External interrupt mask register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	EIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

9-1 SD Card

1. SD Card

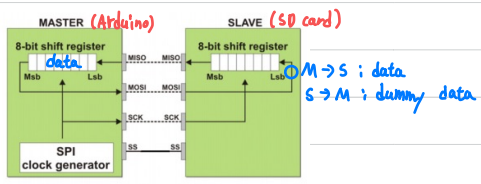
- SD card uses 3.3V voltage level, but Mega is 5V system ⇒ need a level shifter.
- 74HC4050 : Hex non-inverting HIGH-to-LOW level shifter.

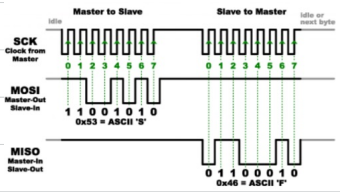


- SPI connection

2. SPI Communication

- Synchronous serial communication used for short distance communication

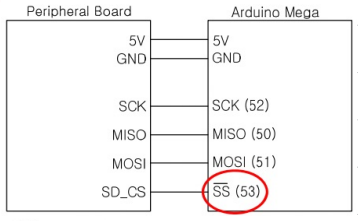




- Single master, multiple slaves.

3. SD Library

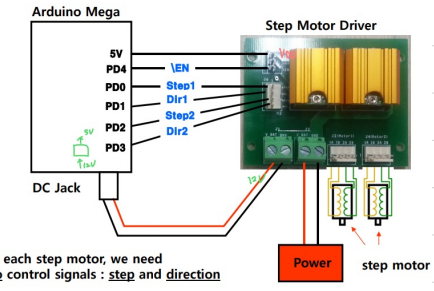
- **Reading** from and **writing** to SD cards
- It supports FAT16 and **FAT32 file systems** on standard SD cards and SDHC cards.
- It uses **8.3 names** for files
  (Ex: rainy.txt (O), rainy_window.txt (x)
  8자 이상
- The communication between the SD cards and the microcontroller uses **SPI**.
  (Ex: **Pins 50, 51, 52, 53 in Arduino Mega**)
- Internal **SPI clock setting** = **4 MHz** (verified through oscillogram)



Line tracer

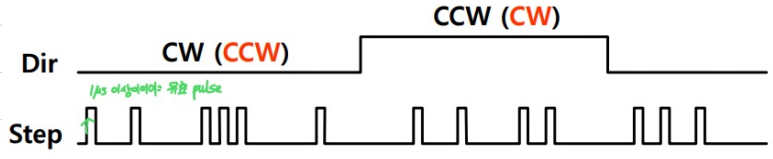
- amount of the current (adjusting current)

⇒ Higher current → higher torque but higher power consumption



For each step motor, we need
two control signals : step and direction

/EN=0 : enabled
/EN=1 : disabled



Each pulse will make the step motor rotate a unit step angle ⇒ $f \propto$ 회전속도

Method	Unit step angle
Full step	1.8 degrees (=360/200)
Half step	0.9 degrees (=360/400)
1/8 step	0.225 degrees (=360/1600) ← Our setting

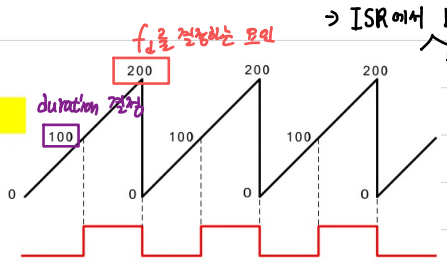
How to make desired freq? : timer interrupt ; 20kHz ⇒ 1초에 20,000 interrupt 걸린다 (f_c : carrier freq)

⇒ ISR에서 Logic H→L 타이밍에서 주기 생성

출력 핀을 f_d를 결정하는 모델

⇒ f_d : desired freq

(Ex) 100 Hz signal



PWM 주기 $\frac{f_{ck}}{(1+Top)} = \frac{20{,}000}{200} = 100\,Hz$

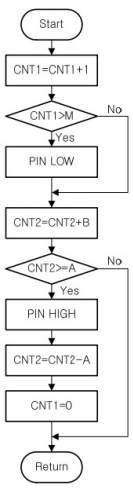
introduce a software counter that is incremented by one in the timer ISR

· DDA(Digital Differential Analyzer)

f_c : carrier freq = 20,000 Hz

f_d : desired freq = 6000 Hz

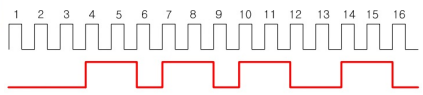
$\Rightarrow \frac{f_c}{f_d} = \frac{20{,}000}{6{,}000} = \frac{10}{3}$ it means generating 3 pulses at every 10 carrier clocks.



$\frac{f_c}{f_d} = \frac{20000}{6000} = \frac{10}{3} = \frac{A}{B}$ $\frac{10}{3}$ 절반 만큼을 High로 가져간다.

$M = \text{int}\left(\frac{A}{B}\right) >> 1 = 1$

Carrier clock	CNT1	CNT2	Pin value
1	1	3	LOW
2	2	6	LOW
3	3	9	LOW
4	4→0	12 → 2	HIGH
5	1	5	HIGH
6	2	8	LOW
7	3→0	11 → 1	HIGH
8	1	4	HIGH
9	2	7	LOW
10	3→0	10 → 0	HIGH
11	1	3	HIGH
12	2	6	LOW
13	3	9	LOW
14	4→0	12 → 2	HIGH
15	1	5	HIGH
16	2	8	LOW

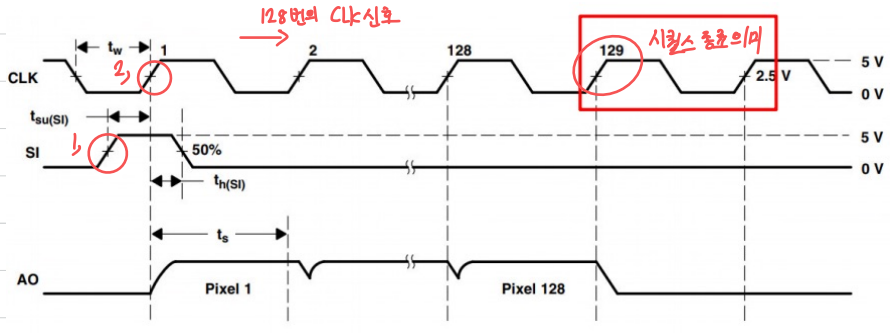


The red signal is 6000 Hz on average.

Simple way : $\frac{f_c}{f_d} = \frac{20{,}000}{f_d} \approx \frac{A}{100}$ (100으로 임의지정) ⇒ $A = \text{round}\left(\frac{20{,}000 \times 100}{f_d}\right)$

Camera

Name	Description
SI	Serial input. SI defines the start of the data-out sequence
CLK	Clock. The clock controls charge transfer, pixel output, and reset.
AO	Analog output → ADC → MC
VDD	Supply voltage. Supply voltage for both analog and digital circuits.
GND	Ground.
AO1	Not used



18번째 Clock cycle 이후부터 다음 SI 신호 까지 시간 t_{int}에 비례하여 AO의 출력 전압 조절

t_{int} ↑

ex, 흰색 : 1.8V 2.8V 3.9V max가 5V이고 폭이 높은 것을 찾으면?

검 : 0.76V 0.96V 1.1V

- ADMUX (ADC Multiplexer Selection Register)
  - Analog reference를 지정 (REFS1, REFS0 이용) 1
  - 사용하려는 ADC channel 지정 (MUX0~MUX5 이용) 0channel 사용
  - 변환 data의 왼쪽 정렬, 오른쪽 정렬 지정 (ADLAR 이용) 오른쪽 정렬
- ADCSRA (ADC Control and Status Register A)
  - ADC Enable (ADEN 이용)
  - Start of conversion을 0으로 clear (ADSC 이용)
  - Auto trigger를 disable (ADATE)
  - ADC interrupt flag을 clear (ADIF)
  - ADC interrupt를 disable (ADIE) 3 사용X
  - ADC Prescaler를 지정(ADPS0, ADPS1, ADPS2 이용)

1023 → 5 V
or 기울기
1023 → 5 V