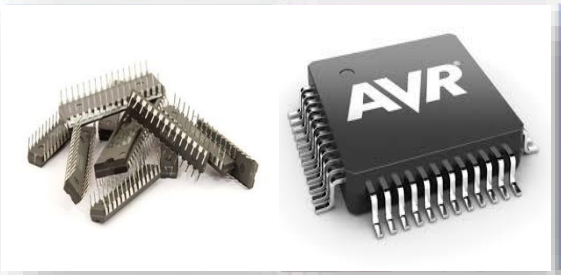
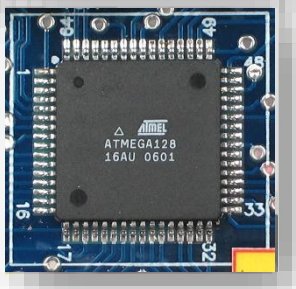
한국공학대학교
TECH UNIVERSITY OF KOREA





10 June 2022

요리용 타이머 동작

1초 계수 방법

- **1클럭의 주기는** $\frac{1}{14.7456MHz}$ **= 67.8**ns

시스템 클럭	14.7465MHz	
분주비	1024	
카운터 입력 클록	0.014400879MHz	
1 클록의 시간	69.44us	
1초 계수를 위한 횟수	14405	
실제 시간	1.000초	

- **69.44us를 기준 클록으로 사용하여 1초를 만들기 위해서는 14405회를 계수하면 (69.44us × 14405 =1.000 sec) 됨.**
- **1차 계수를 5회, 2차 계수를 2881회로 설정하여 소프트웨어 루프를 활용하여 프로그램을 작성함.**
- **타이머/카운터 0의 오버플로우 인터럽트를 발생 주기를 TCNT0의 레지스터 값을 5로 조정하면 69.42μs(67.8ns × 1024) × 5 = 347.1μs 마다 인터럽트 발생되고, 인터럽트의 회수를 2881회 계수하면 약 1초가 됨.**

1초 계수 방법

타이머/카운터0, 일반모드, 분주비 1024, OC0 차단 모드 사용
오버플로우 INT 사용하여 작성

실험 보드의 시스템 클럭은 14.7456MHz 사용, 분주비 1024 사용
1개의 클럭의 주기 = $\frac{1}{14.7456MHz/1024} = 00694[\mathrm{msec}]$
14,400개의 펄스를 계수하면 1초가 됨.
144개의 클럭을 계수하는 루프를 100번 반복하여 14,400개의 클럭을 계숙하는 프로그램을 작성하여 활용

Report

요리용 디지털 타이머의 구현

1. FND와 키패드를 이용하여 타이머 설정 시간 결정 및 사용자 에게 표기
   FND 초기상태는 - - - - 으로 표기
2. 타이머의 설정은 키패드의 M1(A) 버튼을 눌러 모드 진입
   진입시 FND 는 0 0 0 0 으로 표기
3. 원하는 시간을 1초 단위로 기재 최대 9999초
4. 타이머의 시작은 키패드 M4(D) 버튼을 눌러서 시작
5. 타이머의 종료가 되면 FND에 - E n d 를 출력하고 종료.

Code – 1

```
#define F_CPU 14.7456E6
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

unsigned char zero_flag = 0;
unsigned int sec_count = 0;
unsigned char Port_char[] = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xd8, 0x80, 0x90};
unsigned int Port_fnd[] = {0x1f, 0x2f, 0x4f, 0x8f};

void port_Init()
{
	DDRC = 0x0f;
	DDRB = 0xff;
	DDRE = 0xf0;
}

void Init_Timer0()
{
	TCCR0 |= ((1<<CS02)|(1<<CS01)|(1<<CS00));
	TCNT0 = 0x70;
	TIMSK |= (1<<TOIE0);
}

ISR(TIMER0_OVF_vect)
{
	TCNT0 = 0x70;
	sec_count++;
}
```

Code – 2

```
unsigned char keyScan(void)
{
unsigned char key_scan_line = 0xf7;
unsigned char key_scan_loop = 0;
unsigned char getpinData = 0;
unsigned char key_num = 0;

	for(key_scan_loop=0; key_scan_loop<4; key_scan_loop++)
	{
		PORTC = key_scan_line;
		_delay_us(1);
		getpinData = PINC & 0xf0;

		if(getpinData != 0)
		{
			switch(getpinData)
			{
				case 0x10:
				key_num = (key_scan_loop*4) + 1;
				break;
				case 0x20:
				key_num = (key_scan_loop*4) + 2;
				break;
				case 0x40:
				key_num = (key_scan_loop*4) + 3;
				break;
				case 0x80:
				key_num = (key_scan_loop*4) + 4;
				break;
				default:
				break;
			}
			return key_num;
		}
		key_scan_line = (key_scan_line >> 1);
	}
	return 0;
}
```

Code – 3

```
unsigned char changeNum(unsigned char key_num)
{
	unsigned char return_num = 0;

	if(key_num % 4 == 0)
		return_num = 12 + key_num / 4;
	else if(key_num / 4 == 0)
		return_num = (4 * (key_num/4) + key_num % 4);
	else if(key_num / 4 == 1)
		return_num = (4 * (key_num/4) + key_num % 4) - 1;
	else if(key_num / 4 == 2)
		return_num = (4 * (key_num/4) + key_num % 4) - 2;
	else if(key_num / 4 == 3)
		return_num = (4 * (key_num/4) + key_num % 4) - 3;
	else;

	if(return_num == 11)
	{
		return_num = 0;
		zero_flag = 1;
	}

	return return_num;
}
```

Code – 4

```
unsigned char changeNum(unsigned char key_num)
{
	unsigned char return_num = 0;

	if(key_num % 4 == 0)
		return_num = 12 + key_num / 4;
	else if(key_num / 4 == 0)
		return_num = (4 * (key_num/4) + key_num % 4);
	else if(key_num / 4 == 1)
		return_num = (4 * (key_num/4) + key_num % 4) - 1;
	else if(key_num / 4 == 2)
		return_num = (4 * (key_num/4) + key_num % 4) - 2;
	else if(key_num / 4 == 3)
		return_num = (4 * (key_num/4) + key_num % 4) - 3;
	else;

	if(return_num == 11)
	{
		return_num = 0;
		zero_flag = 1;
	}

	return return_num;
}
```

```
void fnd_output(unsigned int final_num)
{
	int buffer = 0;
	unsigned char FND0=0,FND1=0,FND2=0,FND3=0;

	FND3 = final_num / 1000;
	buffer = final_num % 1000;
	FND2 = buffer / 100;
	buffer = buffer % 100;
	FND1 = buffer / 10;
	FND0 = buffer % 10;

	PORTE = Port_fnd[0];
	PORTB = Port_char[FND0];
	_delay_ms(10);
	PORTE = Port_fnd[1];
	PORTB = Port_char[FND1];
	_delay_ms(10);
	PORTE = Port_fnd[2];
	PORTB = Port_char[FND2];
	_delay_ms(10);
	PORTE = Port_fnd[3];
	PORTB = Port_char[FND3];
	_delay_ms(10);
}
```

Code – 5

```
void start_fnd()
{
	PORTE = Port_fnd[0];
	PORTB = 0xbf;
	_delay_ms(10);
	PORTE = Port_fnd[1];
	PORTB = 0xbf;
	_delay_ms(10);
	PORTE = Port_fnd[2];
	PORTB = 0xbf;
	_delay_ms(10);
	PORTE = Port_fnd[3];
	PORTB = 0xbf;
	_delay_ms(10);
}

void end_fnd()
{
	PORTE = Port_fnd[0];
	PORTB = 0xa1;
	_delay_ms(10);
	PORTE = Port_fnd[1];
	PORTB = 0xab;
	_delay_ms(10);
	PORTE = Port_fnd[2];
	PORTB = 0x86;
	_delay_ms(10);
	PORTE = Port_fnd[3];
	PORTB = 0xbf;
	_delay_ms(10);
}
```

Code – 6

```
int main(void)
{
	int key_in_value = 0;
	unsigned char key_num = 0;
	int fnd[4] = {0,0,0,0};
	int start = 0, finish = 0;
	unsigned int timer_start = 0;
	int final_num = 0;

	port_Init();
	Init_Timer0();
	sei();

	while (1)
	{
		key_num = keyScan();
		key_in_value = changeNum(key_num);

		if(key_in_value == 13)
			start = 1;
		else if(key_in_value == 16)
		{
			timer_start = 1;
			start = 0;
			sec_count = 0;
		}
		else;
```

.
.
.
.
.
.
.
.
.

Code – 7

```
if(start == 1)
{
	if((key_in_value >= 0) && (key_in_value < 10))
	{
		fnd[3] = fnd[2];
		_delay_ms(100);
		fnd[2] = fnd[1];
		_delay_ms(100);
		fnd[1] = fnd[0];
		_delay_ms(100);
		fnd[0] = key_in_value;
	}
	final_num = 1000*fnd[3] + 100*fnd[2] + 10*fnd[1] + fnd[0];
	fnd_output(final_num);
}
else if(timer_start == 1)
{
	if(sec_count >= 99)
	{
		if(final_num > 0)
		{
			final_num--;
			sec_count = 0;
		}
		else
		{
			fnd[3] = 0;
			fnd[2] = 0;
			fnd[1] = 0;
			fnd[0] = 0;

			timer_start = 0;
			finish = 1;
		}
	}
	else;
	fnd_output(final_num);
}
else if(finish == 1)
{
	end_fnd();

	if(sec_count >= 500)
		finish = 0;
      else;
}
else if(start == 0)
	start_fnd();
else;
	}
}
```