Hochschule München FK03	Embedded Systems, 90 Minuten		Prof. Dr.-Ing. T. Küpper
Zugelassene Hilfsmittel: alle eigenen, Taschenrechner	**Matr.-Nr.:**	**Name, Vorname:**	
	Hörsaal:	**Unterschrift:**	

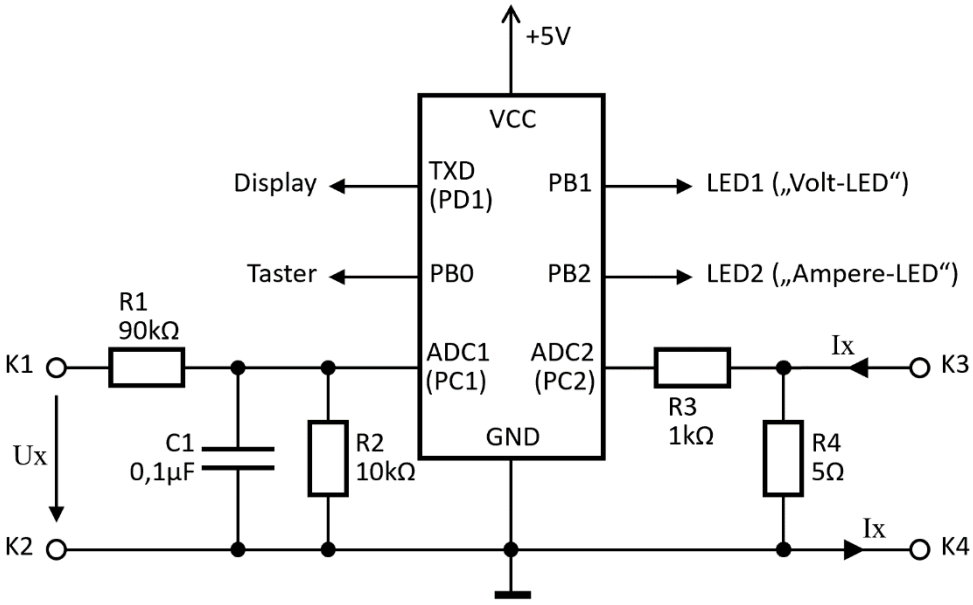
WS 2018/19

Viel Erfolg!!

1	2	3	4	5	6		Σ	N

Aufgabe 1:
Mikrocontroller-Programmierung
(ca. 40 Punkte ≙ 40 Minuten)

Ein digitales Messgerät für Spannungen (an K1 und K2) und Ströme (an K3 und K4) soll mit einem Mikrocontroller des Typs ATmega-328P aufgebaut werden.



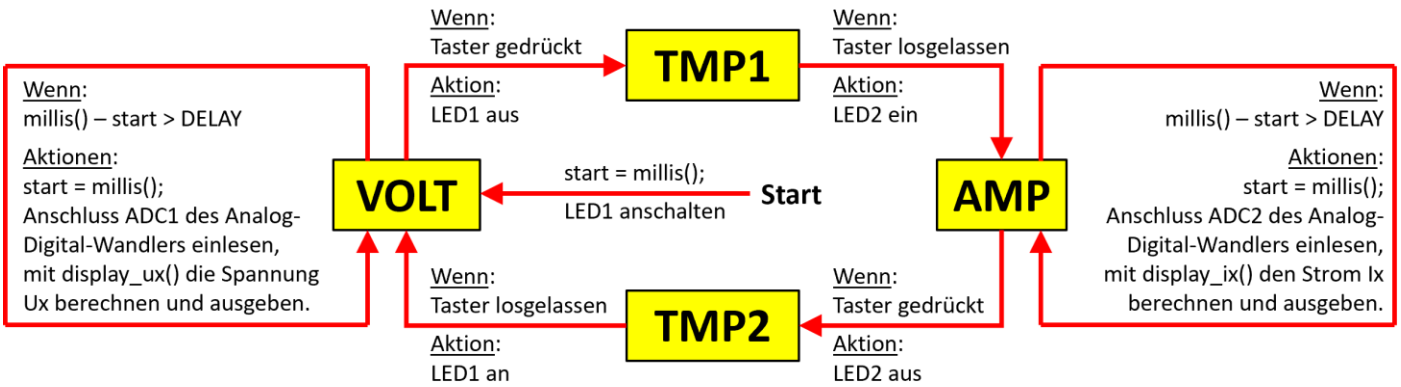
Die Anschlüsse des Mikrocontrollers sind mit den folgenden Komponenten verbunden:

- Anschluss **PB0**:
  Ein Taster zur Umschaltung zwischen den Betriebsarten „Spannung Ux messen“ bzw. „Strom Ix messen“. Der Taster schaltet wie im Praktikum gegen Masse – es wird der interne Pullup-Widerstand verwendet.
- Anschluss **PB1** („Volt-LED“): Diese Leuchtdiode (LED1) leuchtet dann, wenn sich das Messgerät in der Betriebsart „Spannung Ux messen“ befindet. Anschluss **PB2** („Ampere-LED“): Diese Leuchtdiode (LED2) leuchtet dann, wenn sich das Messgerät in der Betriebsart „Strom Ix messen“ befindet.
- Anschluss **ADC1/PC1**: Der Analog-Digital-Wandler verwendet diesen Anschluss, wenn sich das Messgerät in der Betriebsart „Spannung Ux messen“ befindet. Anschluss **ADC2/PC2**: Der Analog-Digital-Wandler verwendet diesen anderen Anschluss, wenn sich das Messgerät in der Betriebsart „Strom Ix messen“ befindet.
- Anschluss **TXD/PD1** (serielle Schnittstelle): An der seriellen Schnittstelle des Mikrocontrollers ist ein Display angeschlossen. Die über die Schnittstelle gesendeten Zeichen werden auf dem Display angezeigt.

Hinweise zur Funktion der Schaltung:

- Für die Spannung U_{ADC1} direkt am Eingang ADC1 des Mikrocontrollers gilt: $U_{\mathrm{ADC1}} = U_{\mathrm{X}} \cdot R_2/(R_1 + R_2)$
- Für die Spannung U_{ADC2} direkt am Eingang ADC2 des Mikrocontrollers gilt: $U_{\mathrm{ADC2}} = I_{\mathrm{X}} \cdot R_4$

Das Verhalten des Messgeräts wird durch den folgenden endlichen Automaten beschrieben:



1.1. Für die korrekte Funktion des Messgeräts muss der „Output Compare Match Interrupt“ genau 1000 mal pro Sekunde ausgelöst werden. Ergänzen Sie die fehlenden Werte in der Funktion init_timer().

1.2. Mit der Funktion millis() kann ermittelt werden, wie viele Millisekunden seit dem Start des Programms vergangen sind. Wozu sind in der Funktion millis() die Aufrufe von cli() bzw. sei() erforderlich?

1.3. Warum muss die Variable millis__intern__ als „volatile“ deklariert werden?

1.4. In der Funktion init_ports() werden die Anschlüsse PB1 (LED1) und PB2 (LED2) als Ausgänge und der Anschluss PB0 (Taster) als Eingang definiert. Zur korrekten Funktion des Tasters wird außerdem der interne Pullup-Widerstand an PB0 aktiviert. Schreiben Sie die dafür notwendigen Befehle in die Funktion init_ports().

1.5. Die Funktion taster() liefert als Rückgabewert eine 1, falls der Taster gedrückt ist. Der Rückgabewert ist 0, falls der Taster nicht gedrückt ist. Schreiben Sie die dafür notwendigen Befehle in die Funktion taster(). Hinweis: Die Funktion taster() soll nicht darauf warten, dass sich das Signal des Tasters verändert – es soll keine Schleife programmiert werden!

1.6. Im Unterschied zum Praktikum beträgt die Referenzspannung des Analog-Digital-Wandlers (ADC) nicht 5,0 Volt sondern 1,1 Volt. Welcher Wert muss in der Funktion init_adc() ins ADMUX-Register geschrieben werden?

1.7. Die Funktion read_adc() kann sowohl dazu benutzt werden, einen Analogwert vom Anschluss ADC1/PC1 einzulesen (Betriebsart „Spannung Ux messen“) als auch dazu, einen Analogwert vom Anschluss ADC2/PC2 einzulesen (Betriebsart „Strom Ix messen“). Der Übergabeparameter „nr“ gibt an, von welchem Anschluss der Analogwert eingelesen werden soll.

Stellen Sie zu Beginn der Funktion read_adc() das ADMUX-Register so ein, dass der Analogwert vom korrekten Anschluss eingelesen wird.

1.8. Wie groß sind die maximale Spannung Ux und der maximale Strom Ix, die von diesem Messgerät korrekt erfasst werden können? Tipp: 1,1 Volt Referenzspannung des Analog-Digital-Wandlers beachten!

1.9. Vervollständigen Sie die Funktion main() so, dass das Programm dem oben gezeigten endlichen Automaten entspricht. Tipp: Zum Einlesen des Analog-Digital-Wandlers verwenden Sie die Funktion read_adc(). Um aus dem Rückgabewert dieser Funktion (im Bereich 0…1023) die Spannung Ux bzw. den Strom Ix zu berechnen und auf der seriellen Schnittstelle auszugeben, verwenden Sie die Funktionen display_ux() bzw. display_ix().

1.10. Programmieren Sie eine geeignete Funktion display_ix(). Tipp: Schauen Sie sich die Funktion display_ux() an!

```
// ----------------------------------------------------------------------------
// Messgerät zur Spannungs- und Strommessung
// ----------------------------------------------------------------------------
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdint.h>
#include <stdio.h>

// Mögliche Zustände des Messgeräts (endlicher Automat)
#define VOLT 1
#define TMP1 2
#define AMP  3
#define TMP2 4

// Geschwindigkeit der seriellen Schnittstelle
#define BAUD 9600
#define MYUBRR F_CPU/16/BAUD-1

// Weitere projektspezifische Definitionen
#define DELAY  500
#define TASTER PB0
#define LED1   PB1
#define LED2   PB2

// Timer initialisieren: 1000 Interrupts pro Sekunde, Aufgabe 1.1
void init_timer(void)
{
	TCCR0A = _BV(WGM01);                                  // CTC-Modus aktivieren

	TCCR0B = ____________________________________________ // Prescaler einstellen

	TIMSK0 = _BV(OCIE0A);                                 // Output Compare Match Interrupt

	OCR0A  = ____________________________________________ // Output Compare Register einstellen

	sei();                                                // Interrupt-System ein
}

// Millisekunden seit Programmstart zählen, Aufgabe 1.3
volatile uint64_t millis__intern__ = 0;
ISR(TIMER0_COMPA_vect) { ++millis__intern__; }

// Millisekunden seit Programmstart abfragen, Aufgabe 1.2
uint64_t millis(void)
{
	uint64_t result = 0;
	cli(); result = millis__intern__; sei();
	return result;
}

// Ein- und Ausgänge festlegen, Aufgabe 1.4
void init_ports(void)
{
	// PB1/LED1, PB2/LED2: Ausgänge, PB0/TASTER: Eingang mit Pullup









}
```

```
// Rückgabe == 1, wenn Taster gedrückt; Rückgabe == 0, wenn nicht; Aufgabe 1.5
int taster(void)
{









}

// AD-Wandler initialisieren, Aufgabe 1.6
void init_adc(void)
{
	// AD-Wandler einschalten, Prescaler = 128 (Datenblatt S. 320)
	ADCSRA = _BV(ADEN) | _BV(ADPS2) | _BV(ADPS1) | _BV(ADPS0);

	ADMUX  = ____________________________________________  // interne Referenz (1,1 Volt)
}

// AD-Wandler abfragen; der Rückgabewert liegt im Bereich 0...1023; Aufgabe 1.7
uint16_t read_adc(uint8_t nr)
{
	// ADMUX-Register einstellen: Bei nr == 1 wird vom Anschluss ADC1
	// eingelesen, bei nr == 2 wird vom Anschluss ADC2 eingelesen.









	// ADSC-Bit setzen, um Wandlung zu starten, das ADSC-Bit wird nach
	// dem Ende der Wandlung automatisch wieder gelöscht.
	ADCSRA |= _BV(ADSC);
	while(ADCSRA & _BV(ADSC)) { }     // Ende der Wandlung abwarten
	return ADC;                       // Ergebnis (0...1023) zurückgeben
}

// Siehe Datenblatt S. 230, USART Initialization (24.6.)
void usart_init(uint16_t ubrr)
{
	UBRR0L = (uint8_t)(ubrr);
	UBRR0H = (uint8_t)(ubrr >> 8);       // Set baud rate
	UCSR0B = _BV( RXEN0) + _BV( TXEN0);  // Enable receiver & transmitter
	UCSR0C = _BV(UCSZ00) + _BV(UCSZ01);  // 8 data bits, 1 stop bit
}

// Siehe Datenblatt S. 231, Data Transmission (24.7.)
void usart_transmit(uint8_t data)
{
	while(!(UCSR0A & _BV(UDRE0))) { }    // Wait for empty transmit buffer
	UDR0 = data;                         // Put data into transmit buffer
}

// Siehe Datenblatt S. 233, Data Reception (24.8.)
uint8_t usart_receive(void)
{
	while (!(UCSR0A & _BV(RXC0))) { }    // Wait for data to be received
	return UDR0;                         // Get and return received data
}
```

```
// Komplette Textzeile über serielle Schnittstelle senden
void usart_transmit_line(const char *text)
{
	uint16_t idx;
	for(idx = 0; text[idx] != 0; idx++)
		usart_transmit(text[idx]);		// Alle Zeichen ausgeben
	usart_transmit(13); usart_transmit(10);  // Zeilenumbruch ausgeben
}

// Spannung Ux (in Millivolt, keine Nachkommastellen) auf serieller Schnittstelle ausgeben
void display_ux(uint16_t adc1)
{
	// Hinweis: adc1 ist der vom AD-Wandler gelieferte Wert (0...1023)
	double r1 = 90000.0, r2 = 10000.0;	// Widerstände R1 und R2
	double u_adc1 = adc1 * 1.1 / 1024.0;	// Spannung an ADC1 berechnen
	double u_x = u_adc1 * (r1 + r2) / r2;	// Spannung Ux (in Volt) berechnen
	double u_x1000 = 1000.0 * u_x;		// Spannung Ux (in Millivolt)

	char buf[16];				// Text ist max. 15 Zeichen lang.
	sprintf(buf, "%5.0f mV", u_x1000);	// Text vorbereiten und...
	usart_transmit_line(buf);		// ...zum Display senden.
}

// Strom Ix (in Milliampere, keine Nachkommastellen) auf ser. Schnittstelle ausgeben, Aufgabe 1.10
void display_ix(uint16_t adc2)
{









}

// Beginn des Hauptprogramms, Aufgabe 1.9
int main(void)
{
	usart_init(MYUBRR);
	init_ports();
	init_timer();
	init_adc();

	// Messgerät beginnt im Zustand "Spannung Ux messen"
	uint8_t state = VOLT;
	uint64_t start = millis();
	PORTB |= _BV(LED1);
```

```
	while(1)  // Hauptschleife
	{
		_delay_ms(100);  // Entprellen des Tasters
		switch(state)
		{
```

```
		}	// Ende der switch-Anweisung
	}		// Ende der while-Schleife
}			// Ende der Funktion main()
```

Aufgabe 2: Mikrocontroller, Speicher (ca. 10 Punkte ≙ 10 Minuten)

2.1. Wodurch unterscheiden sich Mikroprozessoren und Mikrocontroller? Nennen Sie zwei Unterschiede!

2.2. Zur Erweiterung des RAM-Speichers kann bei Bedarf ein externer SRAM-Baustein an den Mikrocontroller angeschlossen werden. Warum ist dies einfacher zu realisieren, als eine Speichererweiterung mittels DRAM?

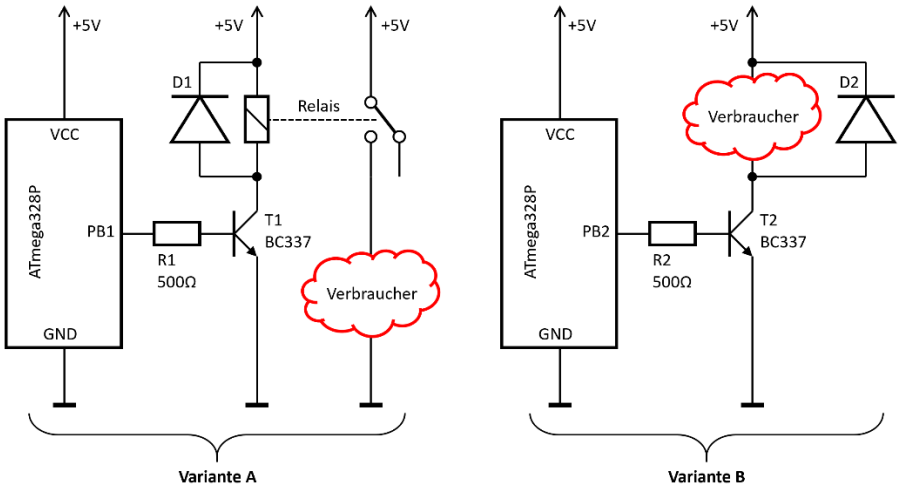
2.3. Zur Speichererweiterung wird ein separater SRAM-Baustein an einen Mikrocontroller angeschlossen. Woher „weiß“ dieser SRAM-Baustein, ob an einer vom Mikrocontroller genannten Speicheradresse Daten geschrieben oder gelesen werden sollen?

2.4. Ein Speicherbaustein ist intern mit 8 Bitleitungen und 8 Wortleitungen aufgebaut. Wie viele Bits können in diesem Baustein gespeichert werden? (Hinweis: An jeder Adresse ist ein Bit gespeichert.)

2.5. Wie viele Adressleitungen sind beim Speicherbaustein aus Unterpunkt 2.4 nach außen geführt?

Aufgabe 3: GPIO-Ports (ca. 10 Punkte ≙ 10 Minuten)

Die linke Abbildung zeigt die Ansteuerung eines Verbrauchers durch ein Relais (Variante A), die rechte Abbildung zeigt die Ansteuerung eines Verbrauchers durch einen einzelnen Transistor (Variante B). Für die Basis-Emitter-Spannung gilt bei beiden Transistoren: U_{BE} = 0,5 Volt. Die Spannung an den Anschlüssen PB1 und PB2 beträgt 4,5 Volt (falls diese eingeschaltet sind). Der Stromverstärkungsfaktor („Großsignalverstärkung“) der Transistoren ist B = 100.



3.1. Welchen Strom I_{PB1} muss der Anschluss PB1 des Mikrocontrollers (Variante A) zum Einschalten des Verbrauchers liefern?

3.2. Welchen Strom I_{PB2} muss der Anschluss PB2 des Mikrocontrollers (Variante B) zum Einschalten des Verbrauchers liefern?

3.3. Welche Schaltung (A oder B) ist nicht geeignet, einen Verbraucher mittels PWM anzusteuern? (Begründung!)

3.4. Ist die Diode D2 erforderlich, wenn In der Schaltungsvariante B als „Verbraucher“ eine Leuchtdiode (LED) angeschlossen ist?

3.5. Kann auf die Diode D1 verzichtet werden und stattdessen die internen Schutzdioden am Anschluss PB1 des Mikrocontrollers verwendet werden? (Sie dürfen in diesem Unterpunkt davon ausgehen, dass die internen Schutzdioden dadurch nicht überlastet werden.)

Aufgabe 4: Grundlagen (ca. 10 Punkte ≙ 10 Minuten)

4.1. Was ist der wesentliche Unterschied zwischen Von-Neumann- und Harvard-Rechnern?

4.2. Der Hersteller des Mikrocontrollers ATmega328P schreibt im Datenblatt, dass dieser Mikrocontroller auf einer „modifizierten Harvard-Architektur“ basiert. Nennen Sie ein Beispiel, wo der Aufbau dieses Mikrocontrollers nicht der „reinen“ Harvard-Architektur entspricht.

4.3. In einem 8-Bit-Prozessor haben viele Register – und insbesondere die Arbeitsregister – eine Größe von 8 Bit. Nur Befehlszähler und Adressregister sind oft deutlich größer (z. B. 16 oder 20 Bit). Warum ist dies so?

4.4. Wie viele Adressleitungen sind mindestens erforderlich, um 1000 Speicherzellen eindeutig adressieren zu können?

4.5. Beim ersten IBM-PC wurde das 4,77MHz-Taktsignal, mit dem damals der Mikroprozessor betrieben wurde, über den sog. ISA-Steckplatz auch an alle Erweiterungskarten weitergegeben. Ein synchroner Betrieb der Erweiterungskarten mit dem Mikroprozessor war dadurch sehr einfach zu realisieren. Warum macht man das heute nicht mehr so? (Beispiel: Beim PCI-Bus wird der Prozessortakt nicht herausgeführt.)

Aufgabe 5: C-Programmierung für Mikrocontroller (ca. 5 Punkte ≙ 5 Minuten)

Geben Sie die Ergebnisse der folgenden Operationen im Binärformat (!) an.

```
uint8_t var1 = _BV(1) | _BV(1);
```

var1 =

```
uint8_t var2 = _BV(1) + _BV(1);
```

var2 =

```
uint8_t var3 = 1 << 2;
```

var3 =

```
uint8_t var4 = 2 << 1;
```

var4 =

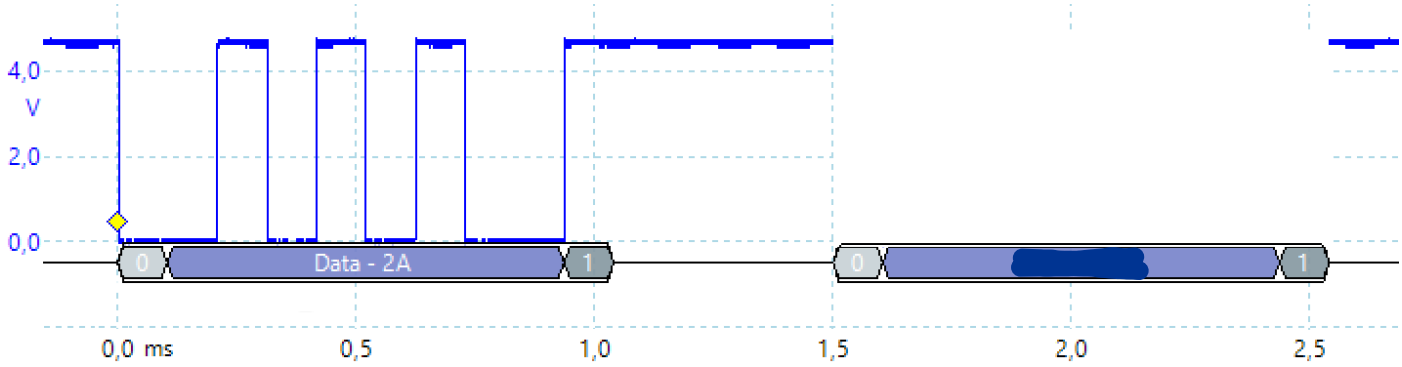
```
uint8_t var5 = 0b11111111 << 2;
```

var5 =

Aufgabe 6: Serielle Schnittstelle (ca. 5 Punkte ≙ 5 Minuten)

Über die serielle Schnittstelle eines ATmega328P wird zunächst das Zeichen * (Stern, ASCII-Code = $2A_{HEX}$) übertragen. Anschließend soll das Zeichen @ (ASCII-Code = 40_{HEX}) gesendet werden. Die Übertragungsparameter sind: 9,6 kbit/s; 8 Datenbits; kein Paritätsbit; 1 Stoppbit.

Zeichnen Sie den Spannungsverlauf, der sich am Anschluss TXD/PD1 (= Ausgang der seriellen Schnittstelle) beim Senden des zweiten Zeichens (@, ASCII-Code = 40_{HEX}) ergibt, in das vorbereitete Diagramm.



Aus dem Datenblatt des Mikrocontrollers ATmega328P

Register: TCCR0B

Name: TCCR0B

Bit	7	6	5	4	3	2	1	0
	FOC0A	FOC0B			WGM02	CS0[2:0]		
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Table 19-10. Clock Select Bit Description

CS02	CS01	CS00	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	clkI/O/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Register: ADMUX

Name: ADMUX

Bit	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR		MUX3	MUX2	MUX1	MUX0
Access	R/W	R/W	R/W		R/W	R/W	R/W	R/W
Reset	0	0	0		0	0	0	0

Table 28-3. ADC Voltage Reference Selection

REFS[1:0]	Voltage Reference Selection
00	AREF, Internal V_{ref} turned off
01	AV_{CC} with external capacitor at AREF pin
10	Reserved
11	Internal 1.1V Voltage Reference with external capacitor at AREF pin

Table 28-4. Input Channel Selection

MUX[3:0]	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5