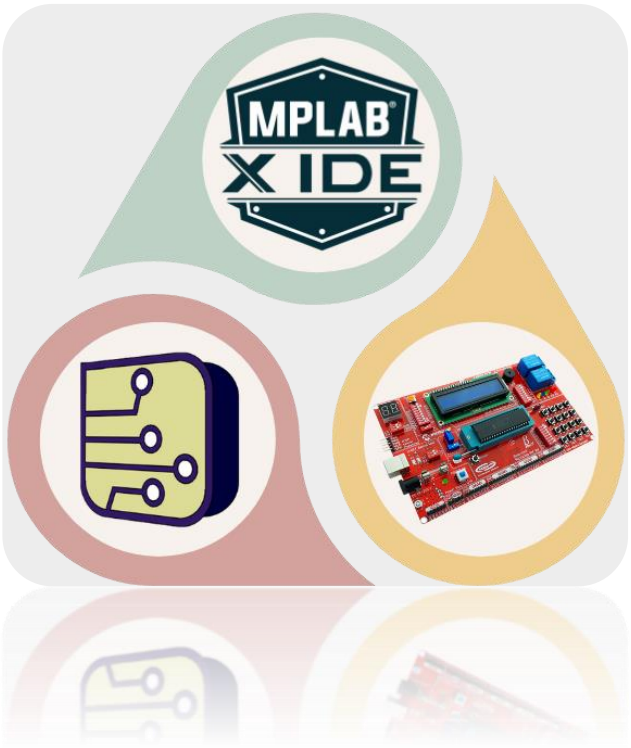
ADC and USART Communication Protocol Using PIC Microcontroller

Practical Lab 4



Objective:

In this Lab we are going to learn how to use the ADC and USART peripherals in the PIC microcontroller and integrate this knowledge in a practical example.

Requirements:

- **Software**:

1. MPLAB
2. Simulide
3. PICSimLab

Experiment Steps:

1. **<u>Open MPLAP IDE:</u>**

   a. Open MPLAP, create a new project.

   b. Create main.c and device_config.c.

   c. Write the functions of LCD, ADC and USART from Tutorial 4.

   d. Take into consideration that the Analog Potentiometer of the bitamini board is connected to Pin A0, So update the adc_init function accordingly.

```
#define _XTAL_FREQ  4000000UL

//#include "device_config.h"
#include <xc.h>
#include <stdio.h>
#include <string.h>

typedef unsigned char uint8;
typedef unsigned short uint16;

#define SET_BIT(REG, BIT_POSN)      (REG |= (1 << BIT_POSN))
#define CLEAR_BIT(REG, BIT_POSN)    (REG &= ~(1 << BIT_POSN))
#define TOGGLE_BIT(REG, BIT_POSN)   (REG ^= (1 << BIT_POSN))
#define READ_BIT(REG, BIT_POSN)     ((REG >> BIT_POSN) & 1)
```

```
/****************** ADC Functions Start ******************/
void adc_initialize(void);
unsigned short adc_read(void);
/****************** ADC Functions End ******************/

/****************** UART Functions End ******************/
void uart_tx_initialize(void);
void uart_rx_initialize(void);
void uart_send(uint8 value);
uint8 uart_read(void);
/****************** UART Functions End ******************/

/****************** LCD Functions Start ******************/
void lcd_4bit_intialize(void);
void lcd_4bit_send_command(uint8 command);
void lcd_send_4bits(uint8 _data_command);
void lcd_4bit_send_enable_signal(void);
void lcd_4bit_send_char_data(uint8 data);
void lcd_4bit_set_cursor(uint8 row, uint8 coulmn);
void lcd_4bit_send_string(uint8 *str);
void lcd_4bit_clear(void);
void convert_uint16_to_string(uint16 value, uint8 *str);
void convert_uint8_to_string(uint8 value, uint8 *str);
/****************** LCD Functions End ******************/
```

e. In main.c we start by initialize each of the LCD, ADC and USART modules.

```
uint16 adc_conversion_result = 0;
uint8 adc_output = 0;
uint8 adc_res_real_txt[7];
void main(void) {
    adc_initialize();
    uart_tx_initialize();
    lcd_4bit_intialize();
    lcd_4bit_clear();

    while (1) {
        adc_conversion_result = adc_read();
        adc_output = adc_conversion_result * 4.88f / 100;
        adc_output = adc_output*3.6;
        uart_send(adc_output);
        convert_uint16_to_string(adc_output, adc_res_real_txt);
        lcd_4bit_set_cursor(1, 1);
        lcd_4bit_send_string(adc_res_real_txt);
    }
    return;
}
```

f. In the main loop, we'll read the value of the adc.

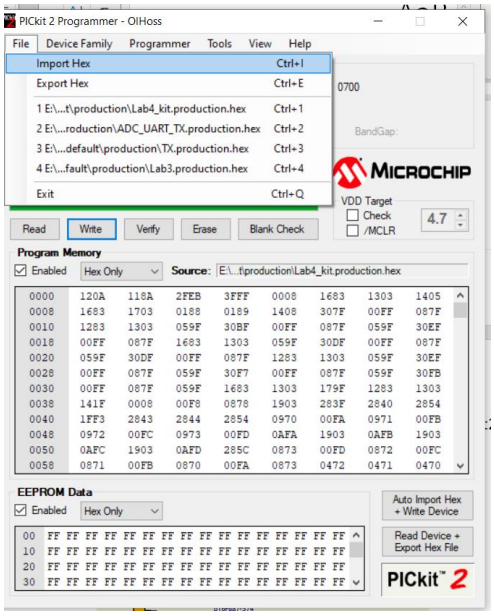
g. Then convert the output value of the adv into voltage value of range (0 to 50) $x10^{-1}$.

h. Finally, we map the voltage value to an angle measure of range (0 to 180). This is done by multiplying the voltage value by (180/50).

i. The value is then send via USART and displayed on the LCD.

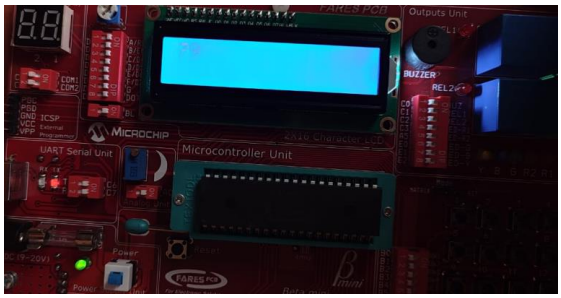
2. **<u>Test the output on the Bitamini:</u>**

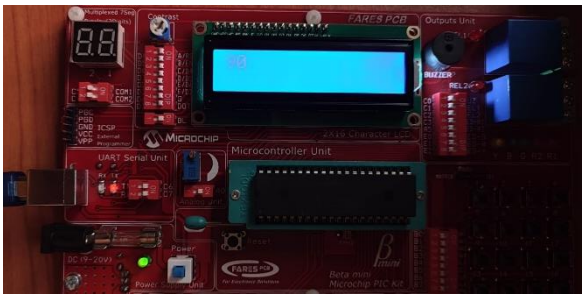
a. Loading your firmware to the kit using PicKit2 software and the PicKit2 Programmer then Press Write to load the code to the kit.



b. Connect the USB Cable to the UART Unit in the Kit.

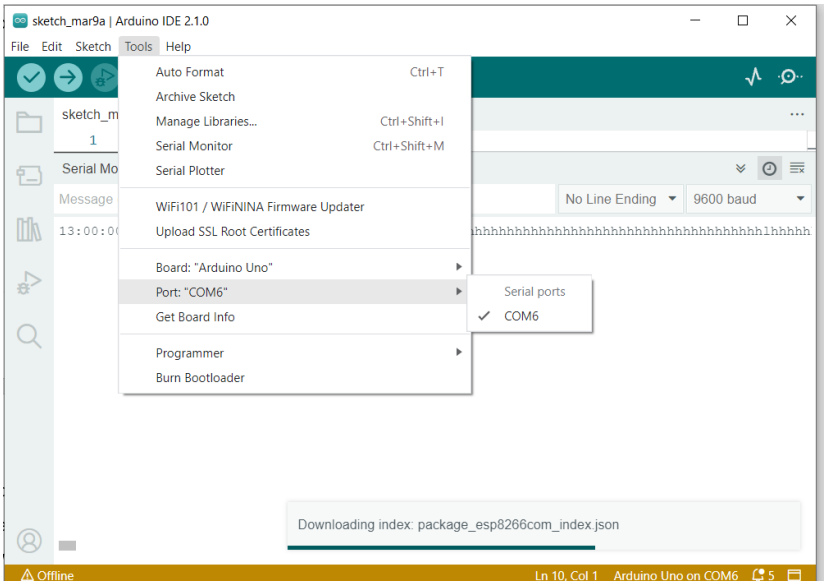
c. The angle value will be shown on the LCD, and by changing the Potentiometer of the analog unite the angle will be changed correspondingly.



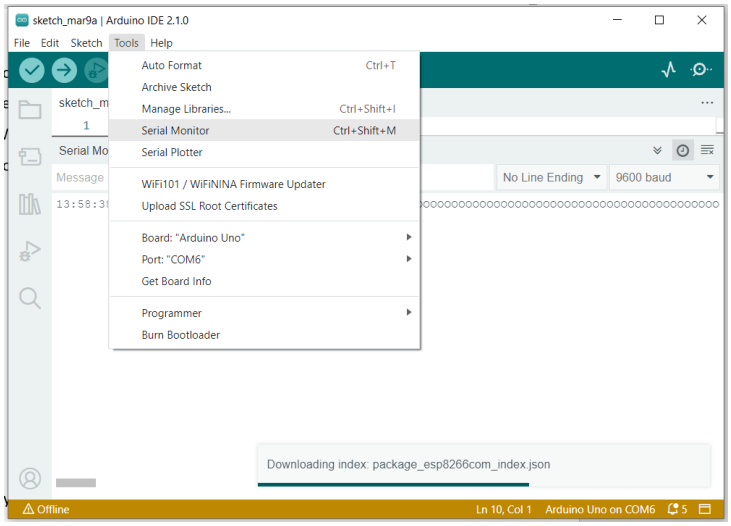


d. To View the serial communication with the PC, we will use the Arduino IDE Serial Monitor.

e. With your kit connected to the PC, Configure the Board and Port from the Tools menu as shown:

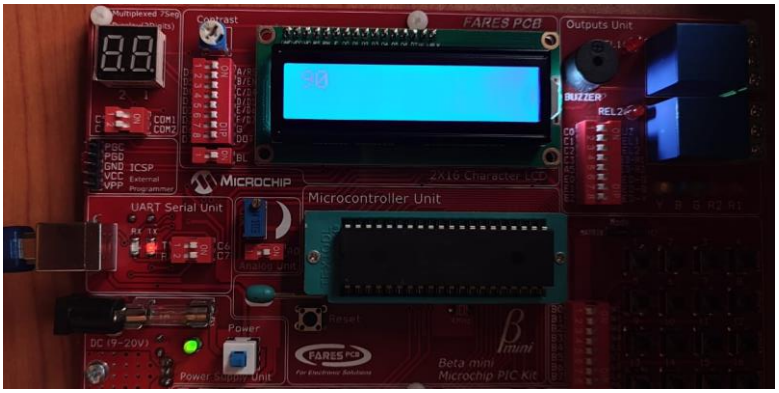


f. Then from Tools click Serial Monitor.

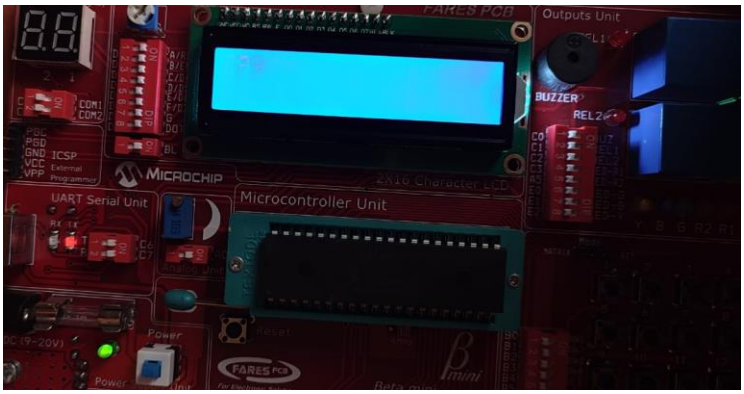


g. By configuring the Baud Rate to 9600 The serial monitor will show the character representation of the decimal value appears on the LCD.

In case of 90



In case of 79

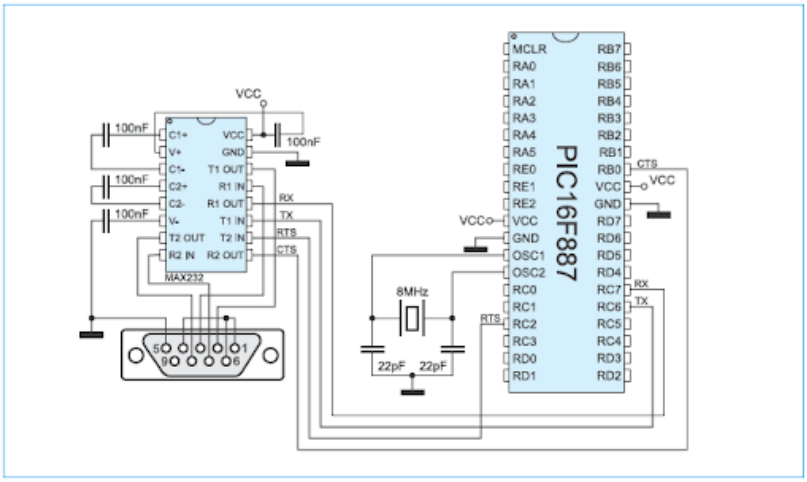


h. You can find the character representation for all available decimal values in the Ascii Table.

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
?	63	0077	0x3f	_	95	0137	0x5f				

3. How the USART Data is sent to the PC?

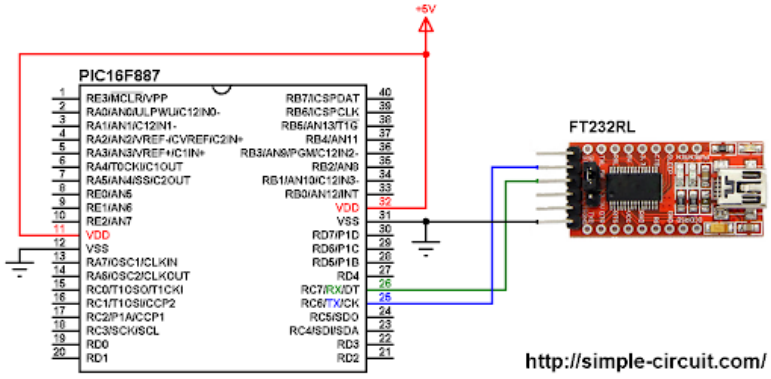
a. The MAX232 is an integrated circuit that is used to convert signals from a serial port to signals suitable for use in TTL (Transistor-Transistor Logic) compatible digital logic circuits. This makes it a key component in serial communications, especially in RS-232 communication interfaces.
RS-232 is a standard for serial communication transmission of data. It commonly uses a DB9 connector with a serial communication port on PCs and other devices, but with the advent of USB, its use has decreased in personal computers. Nonetheless, RS-232 is still widely used in industrial machines, networking equipment, and scientific instruments where a short-distance, low-speed, serial data connection is required.



b. Looking for an alternative to the MAX232 for USB-to-serial communication with a PIC 16F887 (or any other microcontroller), you are essentially moving away from RS-232-based communication to USB-based communication. The MAX232 is specifically designed for RS-232 to TTL serial communication, so when switching to USB, you'll need a different approach. A popular choice for USB communication with microcontrollers is to use a USB-to-UART (Universal Asynchronous Receiver/Transmitter) bridge. These devices convert USB signals directly to TTL serial signals that microcontrollers like the PIC 16F887 can understand.

c. Some common USB-to-UART bridge ICs and modules that are widely used are:

- FT232RL (by FTDI).
- CP2102 (by Silicon Labs).
- CH340 (by WCH).
- Microchip MCP2200.



Lab report:

Submit a PDF file with Code, snapshots and “Small Video for the practical work” of the work you did and upload the project file.