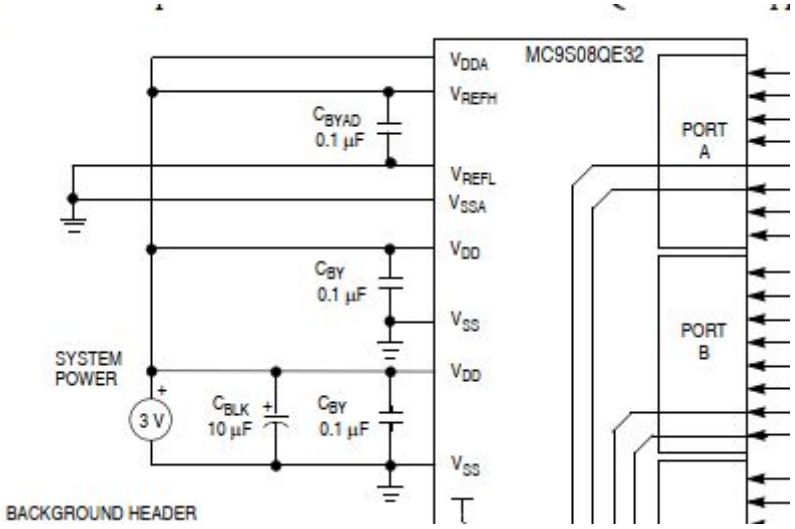
Laboratorio 8: Módulo ADC

Fecha de publicación:
29/05/2019

Fecha de entrega:
12/06/2019

Con esta práctica aprenderemos el uso del módulo ADC, *Analog to Digital Converter*, es un módulo que le permite al MCU leer entradas o niveles de voltaje análogos. (**Importante**: la conversión A/D se hace con referencia a un voltaje alto y bajo que se le configuran por medio de los pines **VREFH**, **VREFL** en el QE16, mucho muy recomendado poner atención a el diagrama de conexiones sugerido en la hoja de datos y la información sobre estas conexiones en el CAPÍTULO.10 del datasheet).



Estas mediciones análogas se convierten a valores digitales codificados en binario por un mecanismo o un circuito interno del que dispone el módulo. De esta manera podemos obtener una aproximación de las entradas análogas.

Envío datos en ASCII por SCI

Antes de tocar el tema del ADC les quiero hablar de algo que me faltó explicarles/pedirles en la práctica de comunicación serial pasada, que pensándolo bien es de mucha utilidad, esto no significa que sea la única manera o siquiera la más óptima, pero es bueno saber de ella y aplicarla alguna vez.
Entonces para iniciar vamos a plantearnos por ahora tres preguntas:

- ¿Cómo enviar datos negativos por serial?

- ¿Cómo enviar datos con punto decimal por serial?
- ¿Cómo enviar caracteres o mensajes por serial?

En la práctica del módulo SCI el acercamiento que le dimos fue envío y recepción de bytes "crudos", literalmente los 8 bits que se escribían correspondian al dato. Ahora se propone codificar el dato a enviar en ASCII.

ASCII son siglas para "American Standard Code for Information Interchange", de esto tenemos que la codificación en ASCII es un estándar para intercambio de información, por lo que hay más *flexibilidad* entre aplicaciones.

Este estándar o la codificación se encuentran fácilmente en internet en tablas como estas:

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Siguiendo esta idea tendríamos que procesar los datos dentro del MCU y codificarlo según el estándar, en el caso de un número, tenemos que asignarle su código a cada dígito y enviar cada uno en orden por el puerto serial. Aquí ya empezamos a hacer un análisis de que puede ser mejor y que no, que puede ser dispendioso o que puede ser más eficiente.

Pongan especial atención a cómo están organizados los datos en la tabla, como ejemplo para la codificación de un número de un dígito basta con sumarle (**48**), y así ya corresponde al valor ASCII según la tabla, EJEMPLO: **Dato**→ 7, **ASCII**→ 7+48=55.

Ventajas:

- Formato y comodidad visual: El uso de comunicación serial el 90% de las veces será para que nosotros podamos visualizar los datos o la información que se envía/recibe, como una forma de *debuggeo*. Con los caracteres especiales como los *Line Finish (10), Tab(9), Carriage Return(13),* o los caracteres como *puntos, comas* y demás, podemos ver de una manera más cómoda los datos en el *interpretador* o la interfaz que recibe los datos.
  Por ejemplo, la lectura de 5 sensores análogos.

```
TERMINAL SERIAL O INTERFAZ
S1:512  S2:123  S3:355  S4:1032  S5:564
S1:543  S2:125  S3:355  S4:1024  S5:234
S1:523  S2:128  S3:344  S4:1032  S5:243
S1:545  S2:156  S3:354  S4:1023  S5:234
S1:573  S2:153  S3:312  S4:1021  S5:223
S1:511  S2:123  S3:353  S4:1013  S5:234
S1:518  S2:113  S3:312  S4:1053  S5:254
S1:591  S2:144  S3:365  S4:1064  S5:212
S1:517  S2:111  S3:351  S4:1012  S5:245
...
```

- Es una codificación estándar: Al ser una codificación estándar, como ya fue mencionado, aporta una facilidad entre programadores o usuarios que utilicen el MCU o el dispositivo, basta con ponerse en el escenario de ser un productor y tener un manual donde diga: *“La información en ASCII debe verse de la siguiente manera….”,* así se establece una facilidad más para el usuario.

Desventajas:

- Costo de procesamiento: En el caso de enviar un número de 32 bits en ASCII nos representa mayor costo de procesamiento (velocidad CPU) que enviarlo en sólo 4 bytes crudos, aqui ya se empieza a jugar con el baud rate del módulo serial.

Módulo ADC: Analog-to-Digital Converter (S08ADC12V1)

A continuación revisaremos los registro y bits para un uso simple y básico del módulo ADC en el QE16.

Primero tenemos esta tabla donde nos presentan todas las entradas o fuentes disponibles de valores análogos.

Table 10-1. ADC Channel Assignment

ADCH	Channel	Input	Pin Control
00000	AD0	PTA0/ADP0	ADPC0
00001	AD1	PTA1/ADP1	ADPC1
00010	AD2	PTA2/ADP2	ADPC2
00011	AD3	PTA3/ADP3	ADPC3
00100	AD4	PTB0/ADP4	ADPC4
00101	AD5	PTB1/ADP5	ADPC5
00110	AD6	PTB2/ADP6	ADPC6
00111	AD7	PTB3/ADP7	ADPC7
01000	AD8	PTA6/ADP8	ADPC8
01001	AD9	PTA7/ADP9	ADPC9
01010	AD10	Reserved	N/A
01011	AD11	Reserved	N/A
01100	AD12	Reserved	N/A
01101	AD13	Reserved	N/A
01110	AD14	Reserved	N/A
01111	AD15	Reserved	N/A
10000	AD16	Reserved	N/A
10001	AD17	Reserved	N/A
10010	AD18	Reserved	N/A
10011	AD19	Reserved	N/A
10100	AD20	Reserved	N/A
10101	AD21	Reserved	N/A
10110	AD22	Reserved	N/A
10111	AD23	Reserved	N/A
11000	AD24	Reserved	N/A
11001	AD25	Reserved	N/A
11010	AD26	Temperature Sensor[1]	N/A
11011	AD27	Internal Bandgap	N/A
11100	—	Reserved	N/A
11101	V_{REFH}	V_{DD}	N/A
11110	V_{REFL}	V_{SS}	N/A
11111	Module Disabled	None	N/A

Pero luego mas adelante nos indican que solo hay un registro para guardar el dato convertido (registro dividido en parte alta y parte baja):

10.3.3 Data Result High Register (ADCRH)

	7	6	5	4	3	2	1	0
R	0	0	0	0	ADR11	ADR10	ADR9	ADR8
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-5. Data Result High Register (ADCRH)

10.3.4 Data Result Low Register (ADCRL)

	7	6	5	4	3	2	1	0
R	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-6. Data Result Low Register (ADCRL)

Entonces, en el QE16 tenemos varios pines para cubrir entradas análogas, pero **solo se puede leer uno al tiempo**; lo que es el mecanismo de adquisición, conversión y luego toma o guardado del dato solo se puede hacer de un canal a la vez. Para leer varios canales toca implementar una lógica que cambie el canal después de una conversión exitosa del canal anterior.

Con esto ya pasamos al primer registro de configuración:

	7	6	5	4	3	2	1	0
R	COCO	AIEN	ADCO	ADCH				
W								
Reset:	0	0	0	1	1	1	1	1

Figure 10-3. Status and Control Register (ADCSC1)

- **ADCH:** Estos 5 bits nos permiten seleccionar el canal a utilizar para hacer la medición.
- **ADCO:** Configuración para habilitar las conversiones continuas si se desea dejar el MCU haga conversiones constantemente o si solo se quiere hacer una conversión en un tiempo dado.
- **AIEN:** Habilitación de interrupción cada que se complete una conversión A/D en el canal seleccionado. El *acknowledge* o la limpieza de la interrupción se realiza cuando se lee el registro ADCRL.

En cuanto a la forma en la que se guardan los datos que se acaban de convertir tenemos dos formas, por **polling (**esperar a que COCO sea igual a 1**)** o por medio de una **interrupción**, con esto ya podemos empezar a entender la magia de las interrupciones en los microcontroladores, en el contexto de una conversión A/D tenemos que el módulo se toma su tiempo, este depende de su configuración, su resolución y a la velocidad a la que trabaje, en resumidas palabras, si lo implementamos por polling podemos estar perdiendo tiempo de procesamiento valioso. Al disponer de una interrupción que ocurra después de que se finaliza una conversión, podemos aprovechar el MCU al máximo.

9.3.7 Configuration Register (ADCCFG)

	7	6	5	4	3	2	1	0
R W	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
Reset:	0	0	0	0	0	0	0	0

Figure 9-10. Configuration Register (ADCCFG)

- **ADICLK:** Selección del reloj fuente para funcionamiento del módulo.
- **ADIV:** Divisor del reloj fuente de entrada del módulo.
- **MODE:** Selección entre 8,10 y 12 bits.

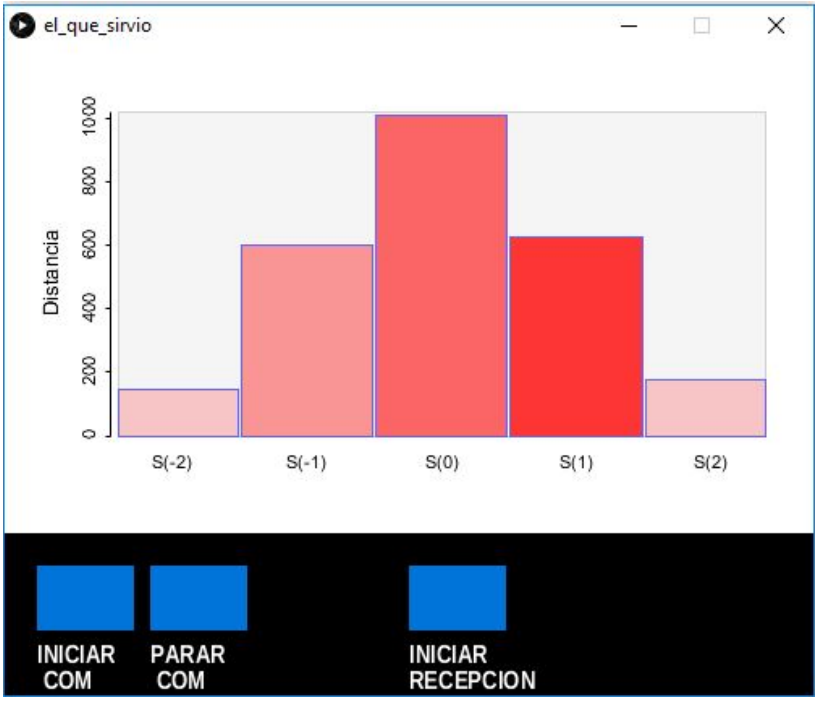
Hay más registros que hacen parte del módulo, estos los deben revisar para que tengan una idea más amplia de las capacidades del módulo en el QE16.

1. **Procedimiento:**
   a. Medir valores análogos de los sensores infrarrojos que utilizaran en el proyecto de curso con una resolución mínima de 10 bits, guardarlos en un arreglo y enviarlos a través del puerto serial en codificación ASCII, el resultado en una terminal o interfaz de puerto serial debe lucir de la siguiente manera:

```
TERMINAL SERIAL O INTERFAZ
S1:512  S2:123  S3:355  S4:1032  S5:564
S1:543  S2:125  S3:355  S4:1024  S5:234
S1:523  S2:128  S3:344  S4:1032  S5:243
S1:545  S2:156  S3:354  S4:1023  S5:234
S1:573  S2:153  S3:312  S4:1021  S5:223
S1:511  S2:123  S3:353  S4:1013  S5:234
S1:518  S2:113  S3:312  S4:1053  S5:254
S1:591  S2:144  S3:365  S4:1064  S5:212
S1:517  S2:111  S3:351  S4:1012  S5:245
...
```

Lo caracteres como “S1,S2...” los espacios y los fines de línea también deben ser enviados desde el MCU.

   b. Se debe implementar una interfaz gráfica que permita visualizar los valores medidos de los sensores infrarrojos que utilizarán para el proyecto de curso, algo por el estilo de lo siguiente:



La idea es que se tenga una línea negra sobre la cual poner los sensores y en la interfaz gráfica se vea como cambian los valores al mover los sensores sobre tal línea.

2. **Preguntas:**
    a. Resumen de la subsección *10.4 Functional Description*
    b. Haga un resumen de lo indicado en la subsección *10.4.4.1 Initianting Conversion.*
    c. Haga un resumen de lo indicado en la subsección *10.4.4.2 Completing Conversion.*
    d. Indique todos los casos de Tiempo de conversión total vs Configuración del módulo (Tabla 10-13 de la sección del módulo ADC).
    e. Determine una relación o expresión que muestre el resultado esperado de la conversión de un valor análogo en un valor binario de 10 bits,
    EJEMPLO: Si entrada_max = 5.0V , ADC→ 1024 , Si entrada_min = 0.0V , ADC→ 0.
    f. ¿Que otra función puede realizar el módulo ADC? (Tip: *10.4.5 Automatic Compare Function*).

3. **Bibliografía:**
    a. Módulo ADC
    b. Datasheet MCU QG8
    c. Datasheet MCU QE16

4. **Tabla requisitos**

Requisito	**Cumple**	**No cumple**
Captura de valores análogos de múltiples canales		
Mínimo 10 bits de resolución en la conversión		
Envío de datos por el puerto serial		
Datos codificados en ASCII		
Visualización de datos enviados en interfaz gráfica.		

5. **Código ejemplo para el laboratorio**
En el contexto entonces del módulo ADC y también de la codificación ASCII les comparto un código donde se hace lectura de dos canales del ADC, se codifican en ASCII y se envían por el puerto serial.

Con esto casi que se les facilita un 60% de la práctica, pero deben ser conscientes primero de que es un código para el ADC en un **QG8** y en la configuración simple de **8 bits** de resolución. Ustedes deben implementarlo en el **QE16**, en configuración mínimo de **10 bits** y con otras opciones de configuración referentes a la velocidad de conversión del módulo (ADIV, ADCLK), y obviamente justificar cada decisión que tomen.

Se implementó un método para leer dos canales consecutivamente uno después del otro, en el ejercicio de esta práctica deben leer más sensores (de 5 a 8) por lo que tal vez el método del ejemplo no sea el mejor, eso también deben analizarlo.

También el método de la codificación en ASCII no es el mejor, pero funciona.