VFE: fondo de escala
G_S → V_o: señal primaria
G_S → V_s: " no adaptada
G_A → V_{ad}: " adaptada

Resumen ADC

(Sensor)

V_{FE_0} = "100°C" [°C, H, Kg, mA, L]
V_{LSB_0} = "0,05°C" resolución

$V_{FE_S} = G_{sensor} \cdot V_{FE_0}$ [V, mV]
$V_{LSB_S} = G_{sensor} \cdot V_{LSB_0}$ [V, mV]

(Acm)

$V_{LSB} = \frac{V_{ref}}{2^n}$

$m_0 = \frac{V_{FE_0}}{V_{LSB_0}}$ G_{sensor} = "80 $\frac{\mu V}{°C}$" (f. transf.)

$m_s = \frac{V_{FE_S}}{V_{LSB_S}} = m_0$ (← cant. de escalones)

$G_{amp} = \frac{V_{FE}}{V_{FE_S}} = \frac{V_{LSB}}{V_{LSB_S}}$

$\mathcal{E}_T = 1\% \cdot V_{ref} = \frac{1}{100} V_{ref} = \mathcal{E}_{ADC} + \mathcal{E}_{tgh} + \mathcal{E}_G = \mathcal{E}_{ADC} + 2\,\mathcal{E}_G$

error de cuantificación → $\mathcal{E}_{ADC} = \frac{V_{LSB}}{2}$

$\mathcal{E}_G = \frac{V_{ref}/2^{n+4}}{V_{ref}} = \frac{1}{2^{n+4}}$

$\mathcal{E}_G = \frac{V_{LSB}}{4}$

$n = \frac{\ln(m_s)}{\ln(2)}$

$R = \frac{V_{LSB}}{V_{LSB_0} \cdot G_{sensor}}$

tiempo de apertura (adquisición)

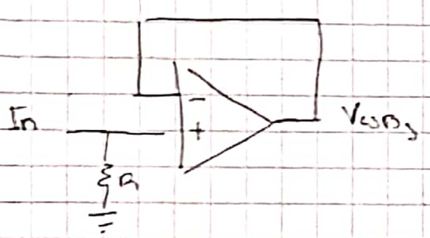
$t_a < \frac{V_{LSB}}{\omega} = \frac{1}{2^n 2\pi f}$

$f_m = 5 f_s = 5 \frac{\omega}{2\pi}$

$\mathcal{E}_T = \pi f\, t_a$

tiempo de conversión mínimo

$t_c = \frac{1}{f_m} = \frac{1}{2 f_s}$ (f_s: señal)



Tolerancia = $\frac{\mathcal{E}_G}{V_{ref}}$ [ppm]

[n] $\cdot 10^6$ [ppm]

Adquisición de datos

8 canales
$f_c = 40\ kHz$
$n = 8$ bits aprox. sucesivas

$AB = \frac{f_c}{n^\circ canales} = \frac{40\ kHz}{8} = 5\ kHz$ Ancho de banda

$f_{mn} = \ldots 2 \cdot f_c$ frec. muestreo mínima

$t_{max} = \frac{1}{N_{can} f_{mn}}$ tiempo del canal máx / " de muestreo

Aprox. sucesivas $t_{max} = (n+1)\, t_{A/D} \rightarrow t_{A/D} = \frac{t_{max}}{n+1} =$ tiempo del clock

$f_{A/D_{min}} = \frac{1}{t_{A/D}} = \ldots MHz$ frec. mínima interna del A/D

Tiempo de conversión

$t_{aprox} = (n+1)\, t_{clk}$ aprox. sucesivas

$t_{d.rampa} = 2^{n+1}\, t_{clk}$ doble rampa

$t_{c_{simple}} = (2^n + 1)\, t_{clk}$ rampa en escalera (simple)

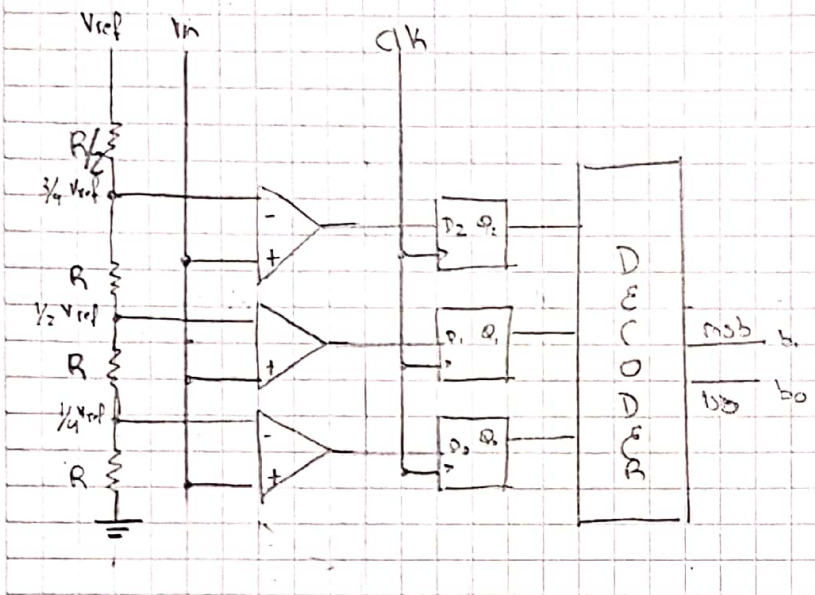
$t_{c_{flash}} = n \cdot t_{clk}$ un ciclo de clock por bit

$t_{max} = t_{conv}$
$t_{A/D} = t_{clk}$

NOTA

A/D Flash (2 bits) $2^n - 1 \rightarrow 2^2 - 1 = 3$ comparadores

Esquema



- El tiempo de conversión está determinado principalmente por el tiempo de retardo de los comparadores (slew-rate), y el t_{max} de propagación existente en el decodificador (10 ns a 100 ns)
- Arquitectura en paralelo
- Comparadores : $2^n - 1$ n = n° de bits
- Consume mucha energía x los comp. a altas velocidades

Diagrama de bloque

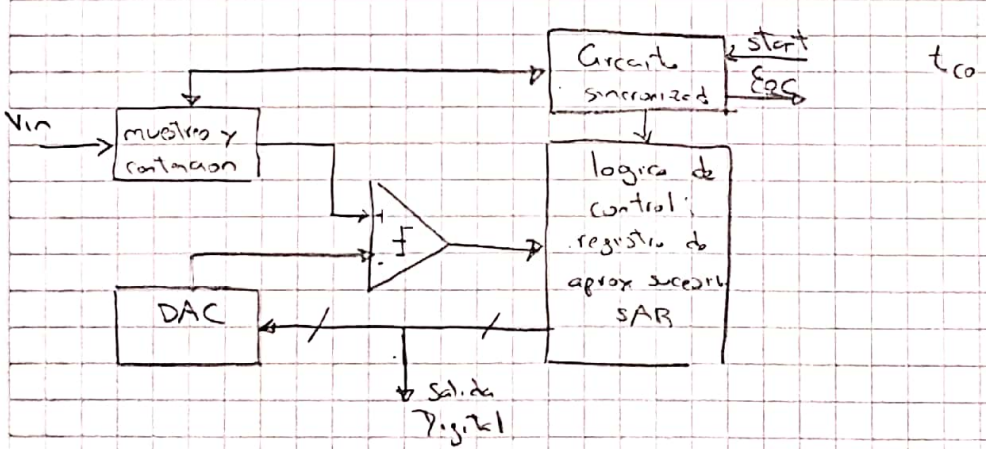
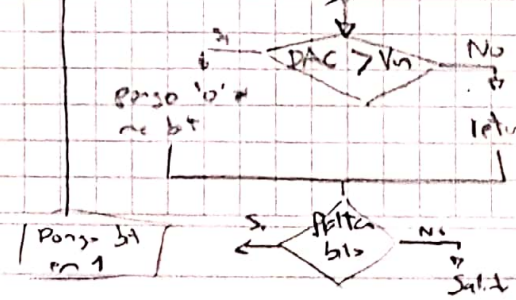


Diagrama de flujo

la señal start coloca al circuito S/H en modo retención y borra todos los bits de la SAR dejando el MSB en 1

la salida de la SAR activa a el DAC interno



NOTA

A/D aproximaciones sucesivas

Registro SAR

A) $\frac{V_{max}}{2^n}$ $V_{in} = 3{,}922\,V$ $V_{max} = 10\,V$ $n = 8\,bits$ $f_{clock} = 15\,kHz$

1° clock → $\frac{10}{2^1} = 5 > V_{in} \therefore$ "0"

2° clock → $\frac{10}{2^2} = 2{,}5 <$ $\therefore$ "1" Acumulado = 2,5V < V_{in} "1"

3° → $\frac{10}{2^3} = 1{,}25 <$ $\therefore$ Acumulado = 3,75 V > V_{in} "1"

4° → $\frac{10}{2^4} = 0{,}625$ Acumulado = 4,375 > V_{in} "0"

5° → $\frac{10}{2^5} = 0{,}3125$ Acumulado 4,0625 > V_{in} "0"

6° → $\frac{10}{2^6} = 0{,}15625$ Acumulado = 3,90625 < V_{in} "1"

B) $t_{clock} = \frac{t_{max}}{n+1}$ → $t_{max} = t_{conv} = t_{clock}(n+1) = 600\,\mu s$

$t_{hold} = 0{,}9 \cdot t_{conv} = 540\,\mu s$

$t_{sample} = 0{,}1 \cdot t_{conv} = 60\,\mu s$

NOTA

RS232

8N1 → 1b start + 1b stop + 0b data

5/02/15

A) · $f_s = 1 kHz$

$f_m = 5 f_s = 5 kHz$

$T = \frac{1}{f_m} = \frac{1}{5 kHz} = 200 \mu s$

$\therefore T = \frac{1}{f_m \cdot 22} = \frac{200 \mu s}{22} = 9{,}09 \mu s$

bit-rate (Veloc. de Transmisión) → $bitrate = \frac{1}{T} = \frac{1}{9{,}09 \mu s} = 110011 bps$

Cantidad de símbolos por segundo → $baud\text{-}rate = \frac{bitrate}{n} = \frac{110011}{8} = 13751 \text{ bauds}$

n: cantidad de bit de datos

→ baud-rate = 9200

2 caracteres = 2 palabras de 8 bit
ej: (8bit + 1 stop + 1start + 1 paridad (?)) = 2(8 bit)

Valores normalizados de bauds

Ironega: 75 110 150 / 300 600 1200 / 2400 4800 9600 / 19200

Profe: 300 1200 2400 / 9600 14400 19200 / 38400 57600 115200

13/10/11

5) Señales

DTR (data terminal ready) : DTE listo para transmitir
DSR (data set ready) : DCE listo para recibir y enviar datos

DCD (data carry detect) : DCE esta recibiendo una portadora (la conexión ha sido establecida con el equipamiento remoto)

RI (ring indicator) : DCE detecto una llamada entrante en la línea telefónica

RTS (request to send) : DTE solicita a DCE transmitir datos
RTR (ready to receive) : DTE listo p/ recibir datos
CTS (clear to send) : DCE listo para aceptar datos

TxD (transmitted data) : canal de envío de datos desde DTE
RxD (received data) : " " " " de DCE

GND (common ground) : referencia a masa

NOTA

7) baud rate: 2400 baudios VPB = Fcrc de reloj

casa de -> DLR = 16 bits
audios
o
gnito
tware

baud-rate = VPB / 16. DLR -> VPB = baud-rate . 16. DLR = 2400 . 16 . 16

VPB = 614,4 kHz

5/07/12

6)

baud-rate = 76800 baudios

baud-rate = VPB / 16. DLR -> VPB = (baud-rate) . 16 . DLR

con DLR = 1 : nivel mínimo

VPB|min = 76800 . 16 = 1,2288 mHz

DLR = VPB / (baud-rate) 16 = 1,2288 mHz / 16 . 9600 = 8 -> DLR = 8 bits

f/baudrate = 9600

7/12/17

4) A) Conexionado entre un DTE - DCE

PIN	DTE		DCE
1	DCD	←	DCD
2	Rx	←	Rx
3	Tx	→	Tx
4	DTR	→	DTR
5	GND	—	GND
6	DSR	←	DSR
7	RTS	→	RTS
8	CTS	←	CTS
9	RI	←	RI

NOTA

RS232

HOJA Nº 2

FECHA

B) Tecnica del checksum:

Puede ser utilizado tanto en la trans sincrona como en la asincrona. Se basa en la transmision, al final del mensaje, de un byte (1 byte) cuyo valor sea el C_2 de la suma de todos los caracteres que han sido transmitidos en el mensaje

El receptor implementara una rutina que sume todos los bytes de datos recibidos y al resultado se le sumara el ultimo byte (que posee la info en C_2 de la suma de los caracteres transmitidos) y si la recepcion del mensaje ha sido correcta, el resultado debe ser cero

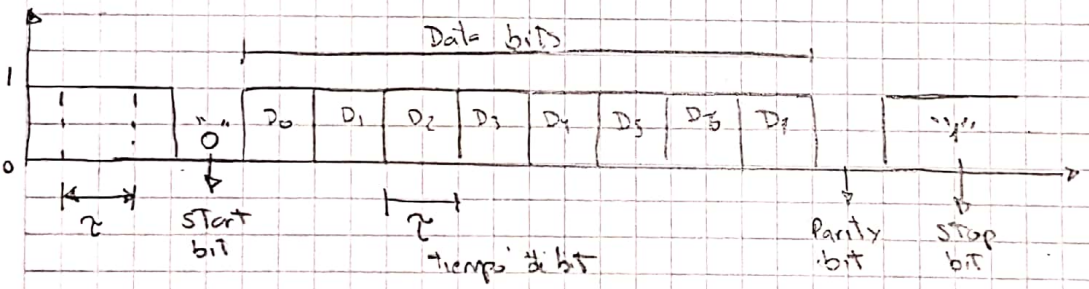
C) Registro DIVISOR (DL → divisor latch)

- El bit 7 tiene como finalidad acceder o habilitar los registros del "latch divisor" tanto en nivel bajo como en alto (DLL y DLM)
- Sirve generalmente p/ setear la velocidad de transmicion. Ej: 9600 baudios

$$\text{Baudios-rate} = \frac{f_{clock}}{16 \cdot DL}$$

23/11/17

4) Trama de la comunicacion serie del protocolo RS-232-C



B) $\tau = \frac{1}{\text{baudio-rate}} = \frac{1}{N} \rightarrow N = \frac{1}{\tau}$ [baudios]

Velocidad de Transmicion = baudio-rate

N = baudio-rate

C) El protocolo RS-232-C es full-duplex porque la comunicacion es bidireccional y simultanea

$Rx \leftrightarrows Tx$

NOTA

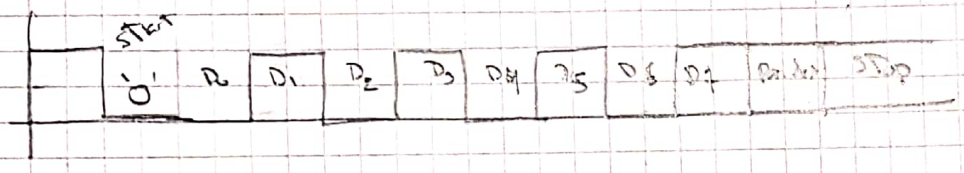
26/06/17

4) N = baudios - rate = 9600 baudios 8N1 → stop + 8 data + start = 10 bits

$\tau = \frac{1}{N} = 104\ \mu s$

max cant de bytes por seg = $\frac{N}{n} = \frac{9600 \text{ baudios}}{10 \text{ bits}} = 960$ [bytes/seg]

5) 801 trama AA = $\overset{D_7}{1}0\,1\,0\,1\,0\,1\,0\overset{D_0}{}$



Con 801 → paridad impar

si cuento la trama AA la paridad es par, pero "le cambia" el bit de paridad

20/12/18

6) Bit de paridad

- Indica si el nº de bits con el valor de "1" en el conjunto de bits de datos es par o impar
- Sirve p/ detectar errores en la transmisión, pero no los corrige si la situación lo requiere, se debe reenviar el dato.

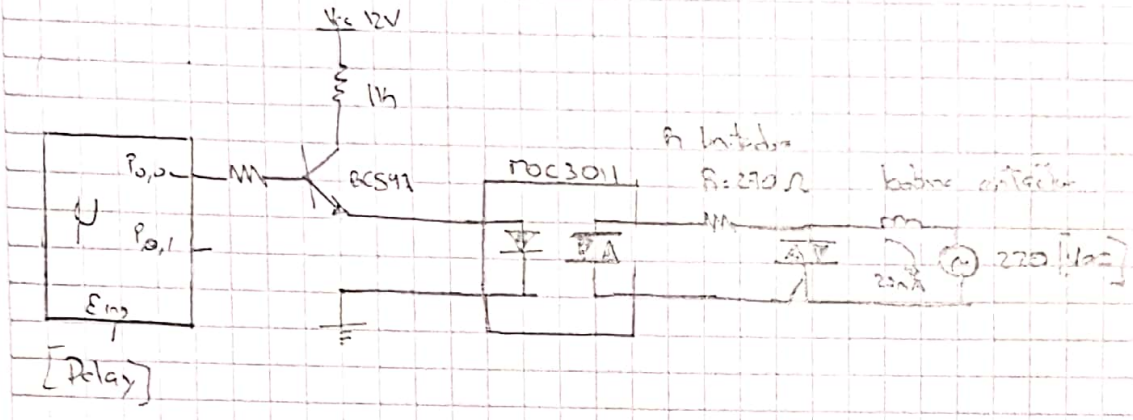
NOTA

Microprocesador

HOJA Nº

FECHA

Interfase entre el procesador ARM y un contactor que se alimenta a 220V y cuya bobina requiere 25mA p/ operar. Debe incluir un optoacoplador



[Relay]

NOTA

Comandos Assembler ARM

mov r0, #100 → copia 100 en el registro "r0"

mov r0, r1 → copia el contenido de r1 en "r0"

ADD r2, r0, r1 → suma el cont de r0 y r1 y lo cobra en r2

ADD r2, r0, #10 → suma el cont de r0 y 10 y lo guarda en r2

Cmp r2, #0 → compara y modifica las banderas no los regis r0 (r2=0)

Sub r2, r1 → resta r2 = r2 - r1

RSB r2, r1 → resta r2 = r1 - r2

Banderas: (lo usa el la toma de decisiones)
- Z (cero) indica si el resultado = 0 Z=1 si el resultado = 0
- N (negativo) sigo de la ultima operación (bit 31) N=1 si el resultado reg.
- C (carry) acarreo
- V (overflow)

ADDS r2, r1, #0 → "S" hace q' modifique las banderas segun el resultado

ADC r → suma con acarreo

SBC → resta con acarreo

TST → Actualiza las banderas con el resultado de AND

mul r0, r1, r2 → multiplica r0 = r1 . r2

lsl, #N → r0 = r1 . 2^n (mult. por 2^n) desplazamientos sin signo

asl → " " " con "

lsr → divide " " con "

asr → " " con "

Mayusc a Mns → mns a may

Add r1, #('a' - 'A') → sub r1, #('a' - 'A')

NOTA

Condicionales (sufijos)

sin signo { > HI ∧ LO < ; ≥ HS ∧ LS ≤ }

con signo { > GT ∧ LT < ; ≥ GE ∧ LE ≤ }

acarreo CS ∧ CC sin acarreo
desborde VS ∧ VC sin desborde
negativo MI ∧ PL positivo o cero
igual EQ ∧ NE desigual

Diferencias for, while, do

for (r0=0; r0<10; r0++) { }	while (r1<10) (cond.) {};	do { } while (r1<10) (cond.)
mov r0, #0	while cmp r1, #10	do while:
for: cmp r0, #10	bHS salir	–
bHS salir	–	–
–	–	cmp r1, #10
–	b while	bLO dowhile
–	salir:	
ADD r0, #1		
b for		
salir: –		
–		

Manejo de variables (manejo de memoria)

mov r1, #pos1 — r1 es un puntero, apunta a una dirección de memoria; pos1 → valor 5; r2 = 5

ldr r2, [r1] — carga el contenido de una posición de memoria en el registro; lee el valor de r1 y lo copia en r2

es la inversa de ldr {
mov r0, #10
STr r0, [r1] — toma el dato de un registro y lo almacena en la posición de memoria; el valor de r0, lo escribe en la pos. de r1 (#pos1)
}

lo mismo p/ str

ldrb r1, [r2] — lee y expande 8 bit // strb
ldrsb r1, [r2] — " " " " con signo // strsb

ldrh r1, [r2] — lee y expande 16 bit // strh
ldrsh r1, [r2] — " " " con signo // strsh

ldr r1, [r2] — lee y no expande ∴ r2 ya debe tener 32 bit // str

lectura de variables

var: .byte 10, 20, 30 — creo una variable tipo byte (8 bit = 1 byte)
varh: .hword 10, 20, 30 — " " con 16 bit — 2 byte
varw: .word 10, 20, 30 — " " " 32 bit — 4 byte
cadena: .ascii "hola bonito" — c/ escribir texto; 1 Byte c/ letra

NOTA

HOJA Nº

FECHA

Load/Store

array[10] = array[5] + y

array[1]	0x14000004
array[0]	0x14000000 →

dirección base

20 = 4x5

ldr r3, [r2, #20] → r3 = array[5] r2 es la dirección base y tiene un desplaz. de 20 (decimal)

r4 = y

add r3, r3, r4 → r3 = array[5] + y

str r3, [r2, #40] → array[10] llevo r3 a la memoria donde me indique (r2 + un desplazamiento de 40 (decimal))

40 = 4x10

Desplazamiento

- Desplazar 3 lugares a la derecha (divido)

$158_{10} = 1001.1110$

$19_b = 0001\ 0011$

$158 / 8 \rightarrow 2^3$

19

158 desplaz.

mov r3, r3, lsr #3

- Desplazar 3 lugares a la izquierda (multiplico)

$158_{10} = 10001\ 1110$

$158 \cdot 8 = 1264_{10}$

$1264_{10} = 1111\ 0000$

mov r3, r3, lsl #3

NOTA