

MICRO-CONTROLLER

ADVANCED

APPLICATIONS

Second Edition

Micro-Controller Advanced Applications

Second Edition

K. Voinowsky

Orchid Systems Inc. | Edmonton | Alberta | Canada

If found, please contact this email: _____

Visit our website:

www.GetToThePoint.Ca

Copyright © 2009

24 July 2009. All rights reserved. We have not entered into the Cancopy agreement. No part of this book may be reproduced or transmitted in any form by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. For information address Orchid Systems Inc. at #304, 10617 – 105 Street, Edmonton, AB, T5H 4P7, CANADA.

Orchid Systems books may be purchased for educational or business use. For information please write: Orchid Systems Inc., #304, 10617 – 105 Street, Edmonton, AB, T5H 4P7, CANADA or contact us by email at mail@OrchidSystems.com.

Copyright Policy Warning

Please note that this publisher will prosecute all copyright violations to the fullest extent of the law in any country. Statutory penalties are \$30,000.00 in Canada, \$50,000.00 in the United States and apply equally to individuals, institutions and corporate entities.

ISBN 978-1-897499-27-6

CONTENTS

References	vii
1. M68HC11 SPI and Voltage Output DAC	1
Problem Exercises	29
Programming Lab Exercise	41
2. M68HC11 Interrupts	65
Problem Exercises	95
Programming Lab Exercise	103
3. M68HC11 Timer Sub-system	117
Problem Exercises	153
Programming Lab Exercise	161

REFERENCES

References throughout the text refer to technical data available from the following sources:

1. M68HC11E Programming Ref. Guide, Rev. 2.1 07/2005 (referred to as: **HC11-RG**; smallest) www.freescale.com/files/microcontrollers/doc/ref_manual/M68HC11ERG.pdf
Quick reference.
2. M68HC11E Family Data Sheet Rev. 5.1 07/2005 (referred to as: **HC11-DS**; mid-size) www.freescale.com/files/microcontrollers/doc/data_sheet/M68HC11E.pdf?fsrch=1
Brief chapters on the various sub-systems, and electrical and timing specifications.
3. M8HC11 Ref. Manual M68HC11RM/D Rev. 6.1 2007 (referred to as: **HC11-RM**; biggest) www.freescale.com/files/microcontrollers/doc/ref_manual/M68HC11RM.pdf?fsrch=1
Most complete and detailed description of operation and instruction set.
4. Analog Devices DAC8512 12-bit DAC: www.analog.com
5. Analog Devices AD5620 / AD5640 / AD5660 12/14/16-bit DAC: www.analog.com

M68HC11 SPI and VOLTAGE OUTPUT DAC

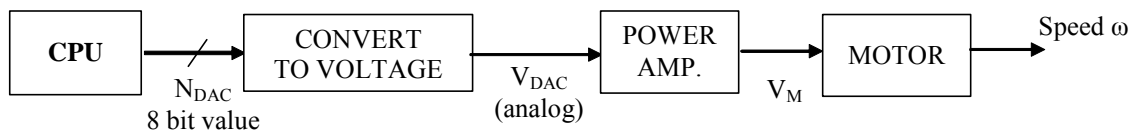
CONTENTS

1. Basic Idea
 - 1.1 Sum of weighted currents and voltages
 - 1.2 Resolution
 - 1.3 R-2R DAC
 - 1.4 Offset
2. Analog Devices DAC8512 12-BIT DAC
 - 2.1 General DAC8512 Features and Specifications
 - 2.2 General DAC8512 Operation
 - 2.3 DAC8512 Digital Side Interface and Timing
3. Analog Devices AD5620 / AD5640 / AD5660
4. M68HC11 SPI
 - 4.1 SPI Interface: Master and Slave Components, Clock Phase and Polarity
 - 4.2 HC11 SPI Pinout & Interface to the DAC8512
 - 4.3 HC11 SPI Software Interface: Control, Status, and Data Registers
5. HC11 SPI Programming
 - 5.1 Initializing the SPI
 - 5.2 16-Bit Data Transmission
6. Example. Generating a Periodic Ramp
7. Example. Allowing User Intervention During Ramp Synthesis
8. DAC Overflow
9. Example. Bi-polar Output Voltage
10. Example. Generating Table Based Periodic Sinusoid
11. More DAC Examples

1. BASIC IDEA

Inasmuch as computers are designed by humans to serve human needs, interaction with them is often via real world, continuous, signals, ex. sound, video, temperature, motion, etc.

→ thus, for example, if we require a computer to control a motor’s velocity, although the computer’s output “command” signal (ex. voltage or current command value) is in a digital form, ex. an 8-bit value between 0x00 and 0xFF, this will have to be somehow converted to an actual physical voltage or current fed to the motor, with a value represented by the computer’s numerical output value, i.e.:



→ thus, we need a device to convert the numerical (“digital”) computer output value, ex. in 8 bits, to a proportional analog voltage value: a Digital to Analog Converter (DAC)

→ this can be easily accomplished if we first understand the mechanism of converting this digital form into a more familiar decimal value: express the computer output value in binary, and then in terms of each bit’s respective weighting. For example consider a simple 4-bit binary value, like 1011_{Bin}:

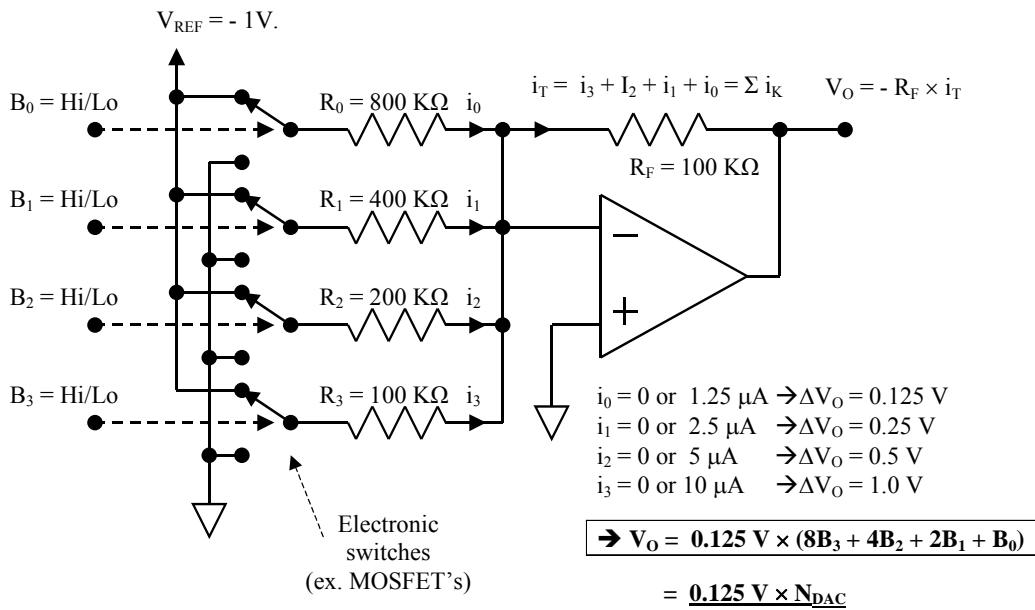
$$\begin{aligned}
 1011_{\text{Bin}} &= B_3 * 2^3 + B_2 * 2^2 + B_1 * 2^1 + B_0 * 2^0 = B_3 * 8 + B_2 * 4 + B_1 * 2 + B_0 * 1 \\
 &= 1*8 + 0*4 + 1*2 + 1*1 = 8 + 0 + 2 + 1 = 11_{\text{Dec}}
 \end{aligned}$$

1.1 SUM OF WEIGHTED CURRENTS AND VOLTAGES

→ thus, if we use each bit to control an electronic switch, ex. a FET, to turn a current on or off, with a value proportional to the respective bit’s weighting, and add all these controlled currents, then we can get a total current proportional to the computer’s output numerical value

→ a simple and straightforward (but not most common) approach is to use a summing amplifier, as shown below, where:

- V_{i3} to V_{i0} are 4 digital voltages, ex. lo or high, each representing, a binary value, and all together: $N_{\text{DAC}} = B_3B_2B_1B_0$
- each of these controls a switch, Sw3 to Sw0, respectively
- each switch connects a reference voltage through a corresponding resistor. NOTE: the relative weighting of the resistors = some reference value × a power of 2
- the inverting amplifier sums the resulting currents, fed through the feedback resistor R_f



→ inverting op. amp. output = weighted sum of the input voltages:

$$V_{DAC} = (B_3 * V_{ref} / R_3 + B_2 * V_{ref} / R_2 + B_1 * V_{ref} / R_1 + B_0 * V_{ref} / R_0) * (-R_F)$$

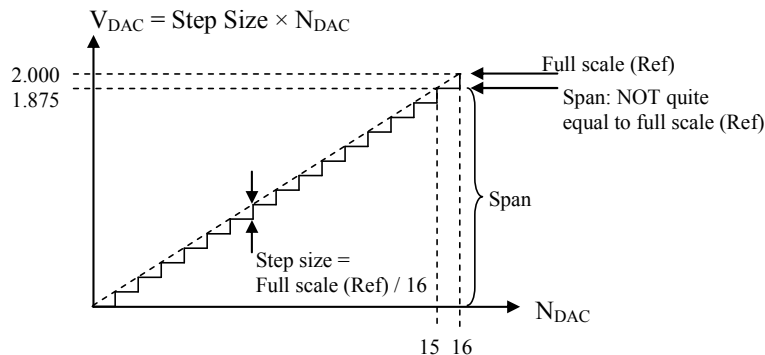
factor out V_{Ref} : $= -R_F * (-1\text{ V}) (B_3 / 100\text{K} + B_2 / 200\text{K} + B_1 / 400\text{K} + B_0 / 800\text{K})$

factor out 800K: $= 100\text{K} * (B_3 * 8 + B_2 * 4 + B_1 * 2 + B_0 * 1) / 800\text{K}$
 $= (B_3 * 8 + B_2 * 4 + B_1 * 2 + B_0 * 1) * 100\text{ K} / 800\text{K}$
 $= N_{DAC} / 8$, where N_{DAC} = the digital value, between 0 and 15
 $= 0.125 * N_{DAC}$

$\rightarrow V_{DAC} = \text{step size} * N_{DAC}$

→ expresses the relationship between the computer output value to the DAC output voltage

→ because the output voltage must be maintained for some time, i.e. until the next sample, the digital input is often supplied from some simple latch, ex. the familiar 8-bit 74HC573 (ex. see address latch in 68HC11 system in expanded mode)

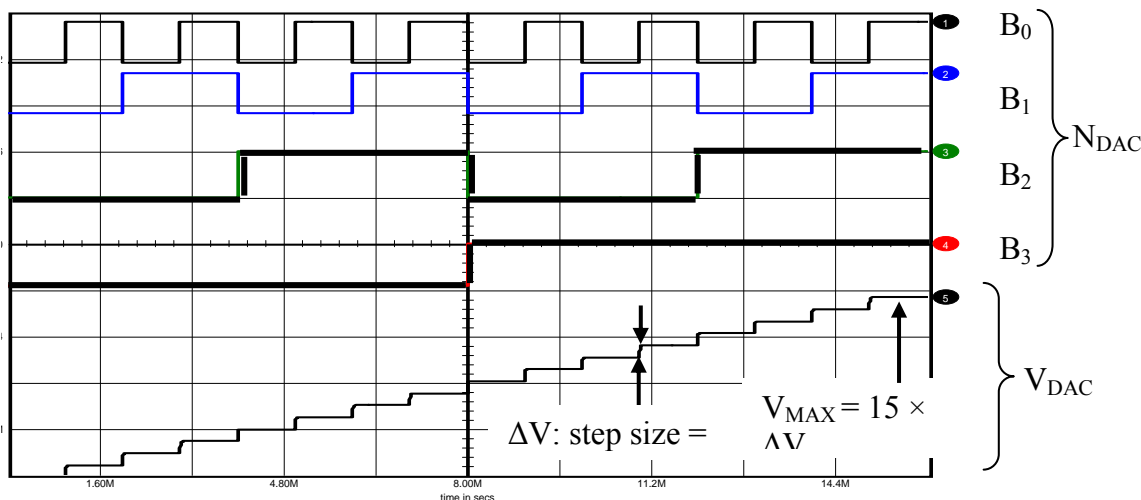


!!! CAUTION !!! not all manufacturers and authors use the same terminology.

→ Main difference: that last one LSBit

ex. compare with 68HC11 ADC (step size = 5 V. / 255 (not 256 = 2⁸))

Ex. consider a simple binary input test signal, consisting of a binary count, from 0000 to 1111, constituting a numerical ramp signal, from 0 to 15_{Dec}, then the DAC output voltage will consist of a ramp voltage, climbing from 0 V. up to 0.125 * 15 = 0 to 1.875 V.:



ex. if the previous DAC is to be extended to 6 bits, then determine:

- i. the values of the required R_4 to R_5 : _____ and _____
- ii. (smallest) step size: _____
- iii. the output voltage span: _____, V_{Min} : _____, and V_{Max} : _____
- iv. NOTE: full scale reference = step size $\times 2^{no. bits} = \underline{8 V.} = \text{span} + \text{last step}$

1.2 RESOLUTION

→ Referring to the above waveform, we can see that the output voltage can only change in discrete steps which are multiples of 0.125 V.

→ this characteristic of DAC's (as well as of A-to-D converters (ADC)) is commonly referred to as the quantization resolution: the smallest change in $V_{DAC} = V_{Span} / 2^n$, n = number of bits.

→ equivalently, this resolution can be expressed as this number of bits n ; thus the above DAC has a 4-bit resolution. This then implies the more meaningful ratio:

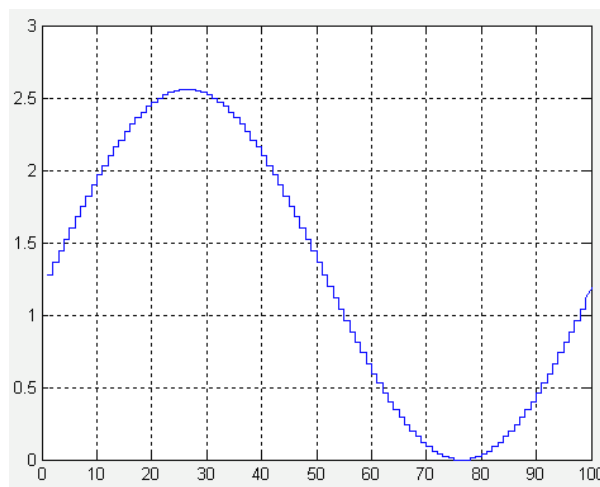
$$\frac{\text{largest signal value}}{\text{smallest signal value}} = 2^{\text{Number of bits}}$$

which can also be alternately expressed in terms of dB, as: $20 \log_{10}(\text{above}) \approx 6 \times \text{Number of bits}$

→ which relates more meaningfully to a measure of signal distortion that can be expected of the converter

ex. thus, for example, a 8-bit DAC, with a full output voltage span of 2.56 V.:

- has a resolution of (!) _____ bits
- has a quantization resolution of:
 $2.56 V / 2^{\text{_____}} = \text{_____ mV.}$



- can produce a full amplitude sinusoid ($2.56 V_{p-p}$) with a “quality” (SNR), or peak to peak amplitude to quantization size ratio = $2.56 V. / 10 mV. = 2$ —
- can produce a full amplitude sinusoid ($2.56 V_{p-p}$) with a “quality”

$$\text{SNR} = \underline{\quad} \times \underline{\quad} \text{ dB/bit}$$

$$= \underline{\hspace{2cm}} \text{ dB}$$

1.3 R-2R LADDER DAC

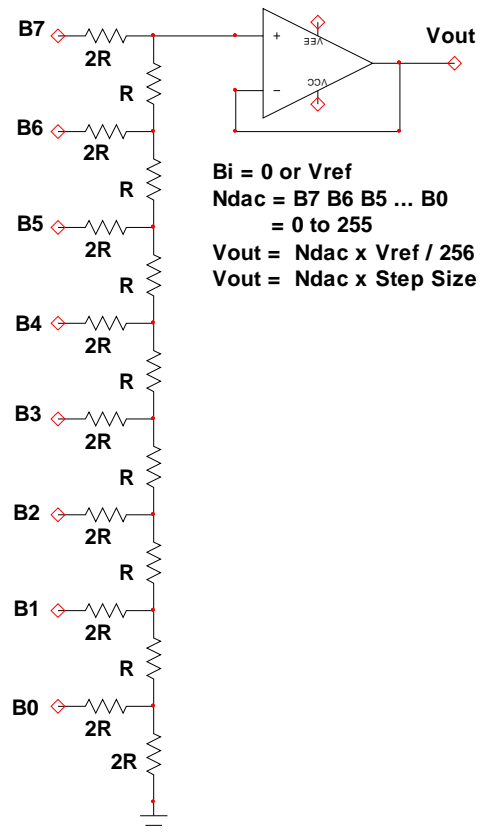
→ the previous DAC circuit, although convenient for a first introduction, is inconvenient from a manufacturing point of view, as it requires resistors with tighter tolerances with finer resolutions: more expensive. Thus, with $N = 10$ bits, the LSBit has a weighting of $1/1024 \approx 1/1000 = 0.1\%$ of the full scale range; the smallest resistor, for the MSBit, must have a tolerance of well under 0.1%.

→ a more common DAC realization consists of a so-called R-2R ladder. This basically amounts to a voltage divider, consisting of resistors with one of only 2 values, ex. 100K and 200K, which significantly reduces the range of values that must be used, and therefore reduces the cost of achieving the required tolerance on each.

→ The ladder is so constructed that each bit’s voltage is divided down by it’s corresponding power of two, and all the bit’s contributions are very elegantly added together.

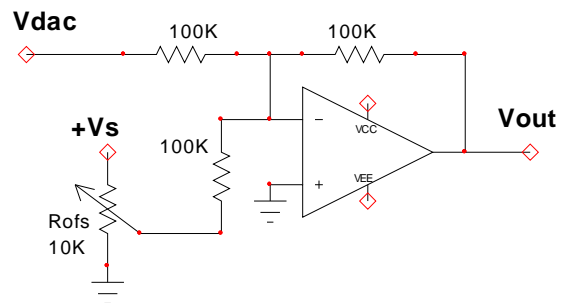
NOTE: regardless of technology, the DAC operation is still described by the relationship:

$$\boxed{V_{\text{DAC}} = \text{step size} * N_{\text{DAC}}}$$



1.4 OFFSET

→ the above circuit produces only a positive output voltage, which may serve well in some applications, ex. a computer controlled unipolar power supply, or to control motor speed in one direction only, whereas other applications may require bi-polar output voltage, ex. audio, motion control (velocity must be negative for position to move “backwards”)



→ this can be accommodated by modifying the previous circuit with a simple offset voltage (adjustable, if necessary), as shown above, which will invert the positive V_{OFS} to the output

→ thus, the output can now be expressed as: $V_{DAC} = 0.125 * N_{DAC} + V_{OFS}$
or, more generally:

$$V_{DAC} = \mathbf{N * Step\ size + Offset}$$

in the same form as the familiar straight line equation: $y = m x + b$

→ the above circuit and equation reveal a number of important issues:

- a reference voltage, which can be seen to determine:
- the output voltage step size, or resolution, here = 0.125 V.
- as well as the total output voltage span, here = 1.875 V.
- if we added more and more bits, of progressively lesser significance, for an ever smaller step size, then the output voltage span would approach 2.0 V., which can often be referred to as the full scale reference voltage, as opposed to the circuit's internal reference voltage

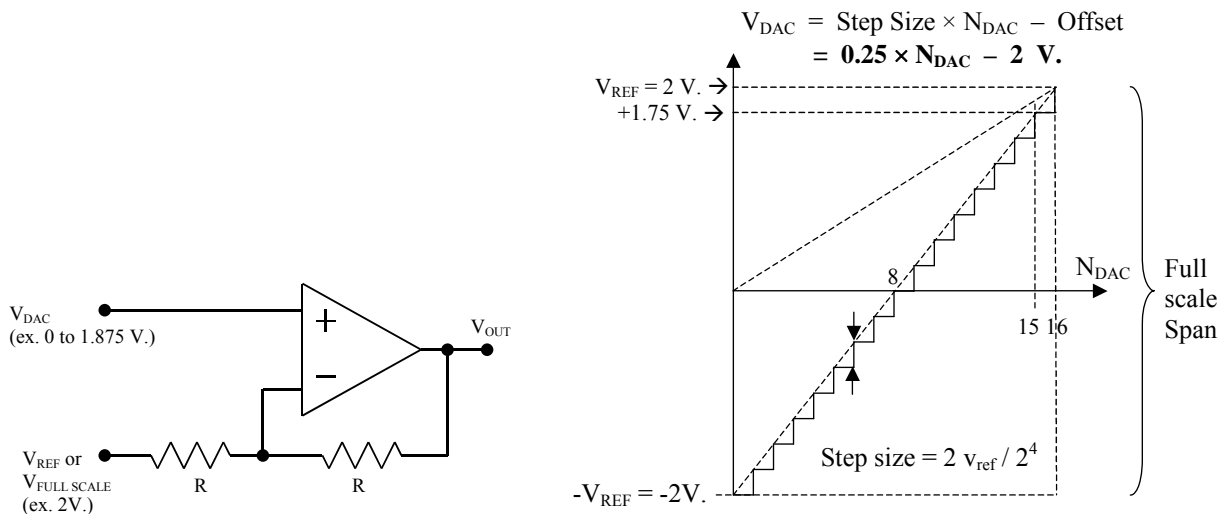
→ thus, we can see:

- output step size = output full scale reference / 2^n , n = no. of bits
- output voltage range = output full scale reference * $(2^n - 1) / 2^n = \text{step size} * (2^n - 1)$

→
$$V_{DAC} = V_{Full\ Scale} / 2^n * N_{DAC} + \text{offset}$$

→
$$\text{Step Size} = V_{Full\ Scale} / 2^n$$

Alternately, it is not uncommon for a DAC device to output the internal reference voltage; ex. see AD5620/40/60. This is very convenient as it simplifies the addition of a simple op. amp. circuit to produce a bi-polar output voltage, from $-V_{REF}$ to $+(V_{REF} - \text{step size})$, as shown below:



→ By linear superposition:

$$\begin{aligned}
 V_{OUT} &= A_{V-NON-INV} \times V_{DAC} - A_{V-INV} \times V_{REF} \\
 &= (1 + R/R) \times V_{DAC} - R/R \times V_{REF} \\
 &= 2 V_{DAC} - 2 V. \\
 &= \text{from } -2 V. \text{ to } +1.75 V. \text{ in steps of } 0.25 V.
 \end{aligned}$$

ex. with 8 bits, if the above circuit's V_{Ref} is changed to $-4V$., and the pot is adjusted for an output offset voltage of $-4V$., then V_{DAC} can be expressed as:

$$V_{DAC} = V_{Full\ Scale} / 2^n * N_{DAC} + \text{offset} = N_{DAC} * 8 / 2^8 - 4\ V. = 0.03125 * N_{DAC} - 4\ V.$$

→ for example, with:

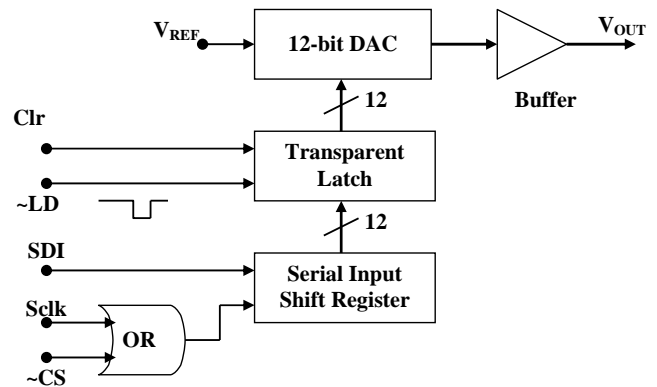
$N_{DAC} = 0_{Dec} = 0x00 = 00000000_2$	→	$V_{DAC} = -4\ V.$
$N_{DAC} = 1_{Dec} = 0x01 = 01000001_2$	→	$V_{DAC} = -3.96875\ V.$
$N_{DAC} = 2_{Dec} = 0x02 = 00000010_2$	→	$V_{DAC} = \underline{\hspace{2cm}}\ V.$
$N_{DAC} = 64_{Dec} = 0x40 = 01000000_2$	→	$V_{DAC} = \underline{\hspace{2cm}}\ V.$
$N_{DAC} = 128_{Dec} = 0x\ \underline{\hspace{1cm}} = \underline{\hspace{2cm}}_2$	→	$V_{DAC} = \underline{\hspace{2cm}}\ V.$
$N_{DAC} = 129_{Dec} = 0x\ \underline{\hspace{1cm}} = \underline{\hspace{2cm}}_2$	→	$V_{DAC} = \underline{\hspace{2cm}}\ V.$
$N_{DAC} = 255_{Dec} = 0x\ \underline{\hspace{1cm}} = \underline{\hspace{2cm}}_2$	→	$V_{DAC} = \underline{\hspace{2cm}}\ V.$

2. ANALOG DEVICES DAC8512 12-BIT DAC

2.1 GENERAL DAC8512 FEATURES and SPECIFICATIONS

→ refer to DAC8512 data sheet; main specs:

- 12 bit resolution
- voltage output
- simple +5V. operation
- internal reference voltage = 4.096 V.
 - simpler external design
 - output resolution = $4.096 / 2^{12}$
= $4.096 / 4096 = 1\ mV$.: convenient
 - $V_{OUT} = 0$ to $4.095\ V$.



NOTE: Serial data input enabled on CS = low

→ Some noteworthy structural details:

- 12 bits in parallel
- 12-bit wide “transparent” D latch, i.e. when LDAC is low, latch is transparent, i.e. output = input, and the output changes as the input changes, and so does V_{OUT} : VERY undesirable; therefore, LDAC should be briefly pulsed low only AFTER all data bits have been shifted in
- Serial In – Parallel Out (SIPO) shift register
- CS input = Chip Select → allows CPU to interface to several external peripherals via serial data and clock, just as via parallel bus, but only the one selected device, i.e. with it's CS activated low, actually receives the data from the CPU

2.2 GENERAL DAC8512 OPERATION

- Serial Peripheral Interface (SPI): very popular, requires very few interface signals to the CPU
 - minimal 8-pin “footprint”
 - one serial data input line
 - synchronous clock input, up to 20 MHz (NOTE: no lower limit on clock speed)
 - data “clocked in”, one bit at a time, into a shift register
 - need to:
 - (a) first, load data bits into shift register, one at a time, in thru **SDI** each data bit is “clocked in”, on rising edge of **CLK**
 - (b) secondly, transfer shift register output, in parallel, i.e. all 12 bits together, into actual DAC circuit input holding data latch, by briefly pulsing LD low
 - see timing diagram below
 - one whole new data sample completely transferred to V_{Out} in under 1 uSec
 - DAC sample rate > 1 MSPS (Mega-Sample Per Sec.)
- this device implements the basic DAC operation described earlier; noteworthy:
 - uni-polar output voltage
 - the DAC binary inputs are held in a 12-bit parallel register, which must first be loaded from a serial input shift register
 - this serial input shift register receives it’s data from the CPU via a 3-wire serial interface, resulting in a conveniently small interface. Although this admittedly results in a reduction in data transmission speed, this is usually accepted. For faster requirements, well then, we need a full width parallel interface, with all the pins for it: a parallel input DAC, also available

2.3 EXAMPLE: ANALOG DEVICES AD5620 / 5640 / 5660 : 12 / 14 / 16 – BIT DAC

Ex. for comparison, same idea as DAC8512, with 1.25 or 2.5 V. voltage reference
 Refer to AD5620/30/40 data sheet:

	AD5620-1	AD5620-2	AD5640-1	AD5640-2	AD5660-1	AD5660-2
Resolution: bits						
V_{REF}						
V_{OUT-FS}						
Resolution: ΔV_{OUT}						

Ex. for further comparison, refer to Texas Instruments’ TLV5630 DAC:

- number of channels: _____
- resolution: _____ bits = _____ mV.
- V_{REF} = _____ V.
- Number of pins: _____

2.4 DAC8512 DIGITAL SIDE INTERFACE and TIMING

→ Refer to DAC8512 data sheet, fig. 1 – Serial digital interface timing diagram (timing requirements); observe:

- (1) before data can be loaded in, CS pin must be activated: _____
- (2) because of internal transparent parallel latch operation, LD must be de-activated: _____
- (3) data comes in, one bit at a time on SDI, starting with: _____
- (4) SCLK (Serial CloCK) loads in each new bit, one at a time, on _____ edge
- (5) after last bit has been sent, all 12 bits must be “transferred”, i.e. loaded, from the shift register into the parallel latch for actual DAC output, by pulsing the LD pin: _____

NOTE: if more than 12 bits are clocked in (more than 12 SCLK cycles), then the register simply retains the _____ 12

→NOTE: the DAC8512 timing diagram only specifies minimum times to be respected, which therefore allows them to be as long as we wish. For example, the SPI signals (SDI, SCK, and LD) could be generated manually, i.e. using switches, for quick, simple, verification of proper DAC operation on a breadboard.

3. M68HC11 SERIAL PERIPHERAL INTERFACE (SPI)

- Referring to the 68HC11 block diagram (HC11-RG, front page):

→ the SPI unit is accessible via **Port** _____, **pins** to _____ (the SCI to RS-232 to PC on pins: _____)

3.1 SPI INTERFACE – MASTER AND SLAVE, CLOCK PHASE AND POLARITY

→ because of the popularity of the “SPI” interface, which requires a (near) minimum of pins to transfer data to a device, many micro-controllers include such a communication port

→ this SPI interface defines two data “terminals”

- a “master” device, which generates the data synchronizing clock signal
- a slave “device”, which receives this data synchronizing clock signal

→ NOTE: this terminology is NOT defined w.r.t. which end sends or receives the data

→ the HC11 can be set up to operate as either master or slave of a SPI connection with a peripheral device. Indeed, this SPI interface can also be used to connect two or more HC11 devices in a multi-processing system.

→ Refer to HC11-RG, p. 49, fig. 7: clock polarity and phase of the 68HC11 SPI unit can be specified via two bits: CP____ : bit _____, and CP____ : bit _____ in the SPI control register (SP____), to select one of 4 clock-data timing relationships, characterized by clock idle condition and clock edge relative to respective data bits

→**QUESTION**: of the four clock timing possibilities in fig. 7, which best agrees to the DAC8512 requirements?

→ CP_____ = _____, CP_____ = _____

3.2 68HC11 SPI PIN OUT & INTERFACE TO THE DAC8512

→ the 68HC11 supports a SPI interface on it's port ____ pins ____ to ____:

- **bit ____ : ~SS** : slave select input or output, like a chip select, enables SPI data transfer to the selected/enabled device
- **bit ____ : SClk** : serial clock signal, out/in, if HC11 set up as master/slave
- **bit ____ : MOSI** : serial data, out/in, if HC11 set up as master/slave
- **bit ____ : MISO** : serial data, in/out, if HC11 is master/slave (none from DAC8512)

→ these pins are operated by the SPI as described above only when the SPI is enabled, which obviously makes them unavailable for general purpose Port D I/O. ALSO, Port D directions must be suitably specified via DDRD, according to each pin's signal direction.

→ clock phase and polarity: a shift register consists basically of D flip flops, and the serial input data is fed into the first such D flip flop. Recall that a D flip flop's operation is basically to "sample" the D input on the rising or falling clock edge. Which edge depends on the actual device, and for the HC11 SPI, this can be programmed to suit the slave device characteristics.

→ referring to the DAC8512 data sheet, fig. 1:

- the clock signal "clocks" the data in on the _____ edge
→ data bits are expected to change and settle first, before the _____ edge of the clock
- the clock signal is shown to "idle" _____, remain high when no data is being sent
- after the data bits are all "clocked in", the serial shift register contents is actually "transferred over", i.e. loaded into the DAC parallel register by pulsing the LD line _____ and back up _____

→ thus, this important diagram defines the complete signal interface, and therefore, what is expected out of the HC11 SPI to properly communicate with the DAC8512

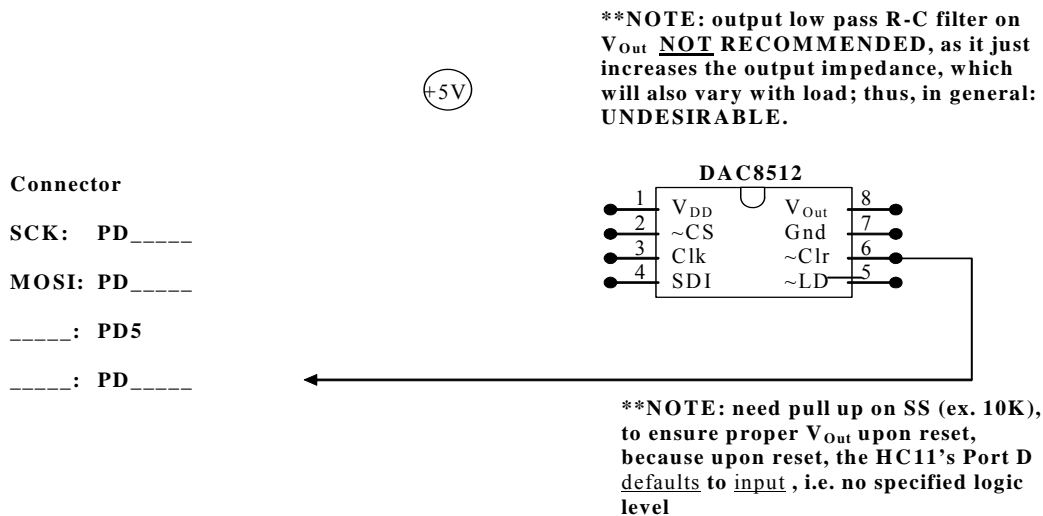
→ considering these two ends of the interface, the DAC8512 can be connected to the HC11 as follows:

- position the DAC physically close to physical V_{OUT} pin, to minimize noise pickup (VERY pertinent, considering 1 mV DAC resolution, and MUCH stronger power supply noise (check it out on the scope)
- tie DAC8512 V_{DD} on pin _____ to _____, and GND to _____, with 0.1 μ F filter cap
- permanently enable the interface: tie CS, pin ____ **to:** Gnd / V_{cc} (which one?)
- **bit ____ : MOSI** : to the DAC8512's _____, on pin ____
- **bit ____ : Sck** : to the DAC8512's _____, on pin ____
- **bit ____ : ~SS** : to the DAC8512's _____, on pin _____, with pull-up resistor
- DAC V_{out} to V_{DAC} vector pin

→ furthermore, we can optionally connect the DAC's ~CLR pin to the CPU's Reset signal, thus clearing the DAC output voltage to 0 upon reset

→ with this suggested connection, after the data has been sent over from the HC11 to the DAC, the DAC's LD pin will have to be pulsed low

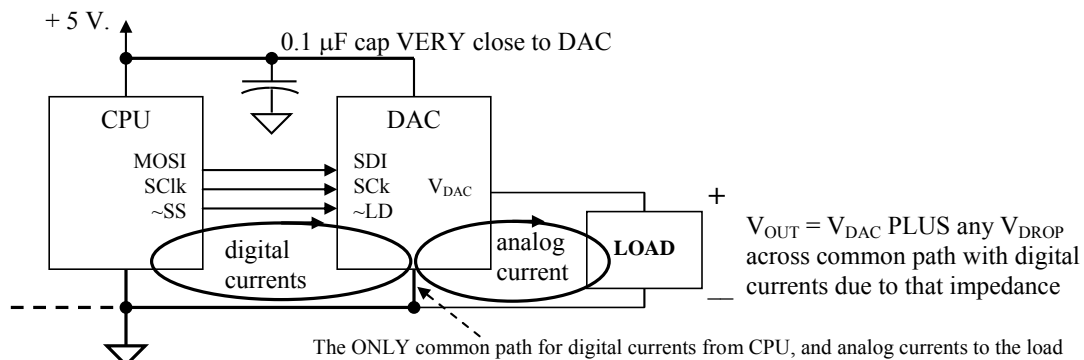
→ the previous information on the DAC8512 and the 68HC11 SPI suggest the following connections:



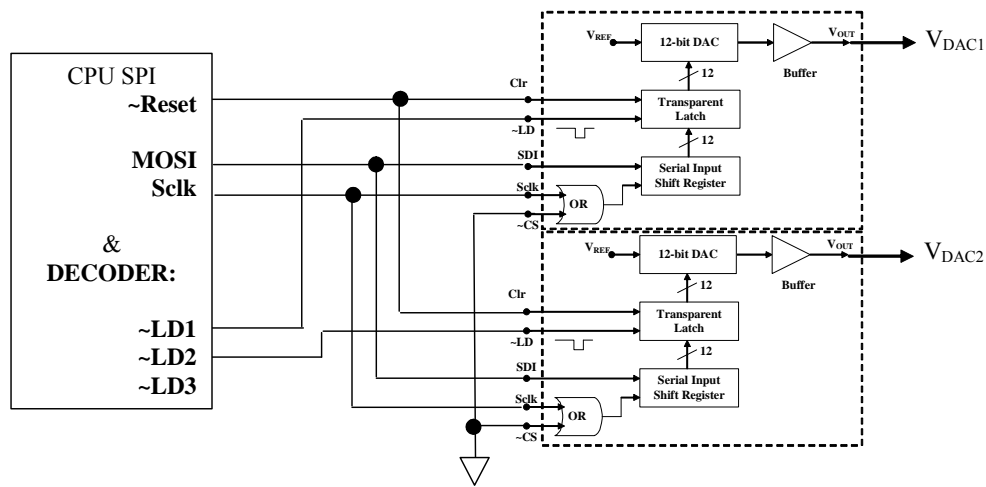
→ Physical connection of DAC to CPU

Very important: the DAC usefulness is defined by its output voltage signal and its quality, defined by a resolution of 1 mV. However, even a casual inspection of the power supply voltage may reveal noise levels in excess of 0.5 V. peak, depending on power supply filtering capacitors and PCB layout, due primarily to very fast switching digital signals, which can completely ruin the quality of the DAC output signal. Therefore, it is very important to plan a physical component and wiring layout to ensure minimal coupling of the any noisy signals to the DAC.

→ important wiring guideline: since the ground is the 0 voltage reference point for all the circuit, i.e. all currents return to that point through whatever path they find, to minimize any cross-talk, i.e. coupling, or interference, between signals, especially digitally switched currents and analog currents such as from a DAC (or to a ADC), it is important to minimize or avoid any common path for these different currents. It is most important to realize that any trace or wire, no matter how short, always has some (A) resistance and (B) inductance, and these can result in significant voltage drops with (A) large currents, ex. through a motor, and (B) fast switching currents. Thus, ideally, the only ground reference point common to all currents should be a single point (ex. see example application notes in various data sheets for high power voltage regulators)



- SUGGESTION: wire the DAC AFTER SPI library code has been written (later on) and the initial SPI test signals have been verified
- If more than one DAC are required, since $\sim LD$ actually “updates” the DAC output voltage V_{Out} , the above principle can be extended or modified for multiple DAC’s, as follows:



3.3 68HC11 SPI SOFTWARE INTERFACE: CONTROL, STATUS, AND DATA REGISTERS

→ referring to the 68HC11 Technical Data Manual (**HC11-DS**):

- the HC11’s SPI clock and data signal timing relationship can be programmed for one of 4 meaningful combinations, according to:
 - clock signal idle state: “clock polarity”
 - clock edge relative to data bit interval: “clock phase”
- NOTE: the 68HC11 SS signal applies only to the HC11 operating in slave mode. For interfacing with a DAC8512, the SS pin can be used to operate the LD pin.

→ Operating the 68HC11 SPI: Control and Status Registers

→ refer to **HC11-DS** fig. 8-1, and observe:

- the SPI derives its clock from the CPU E clock at $f_{XTAL} / 4 = 1.2288$ MHz
- the SPI data speed is set by the SPI clock, determined by the divider, and selected by bits _____ and _____ in the _____ register. For DAC output, the fastest SPI data rate is preferable.
- the SPI operation (HC11 Master or Slave, data pin directions) is determined by bits in the _____ register
- SPI data come out of / in to one common data shift register, _____ bit first
- Data transmission & reception occur simultaneously, i.e. FDX, thru the same shift register
- Completion of data transfer (Tx/Rx) is indicated by _____ bit = 1 in the _____ status register

→ the principal registers involved in operating the SPI are:

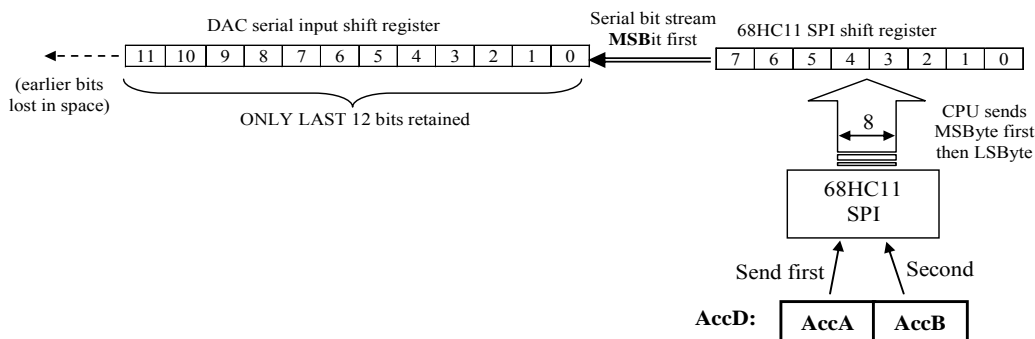
SPCR:	SPIE	SPE	DWOM	MSTR	CPOL	CPHA	SPR1	SPR0
Values:								
SPSR:	SPIF	WCOL		MODE				

Port D:			~SS	SCLK	MOSI	MISO		
Directions:	X	X					X	X
DDRD:								

- the SPI operation is enabled by the _____ bit = _____
- Port D pins _____ are then committed to SPI interface operation
- if the HC11 is to generate the serial clock, it must be: _____: bit _____ = _____
- reconciling timing requirement diagrams for the DAC8512 and the 68HC11 SPI requires:
 CPOL = _____ and _____ = _____
- for 68HC11 to drive output signals actively high or low requires: DWOM = _____
 (alternately, DWOM provides open drain operation for logical Wire-Or connection)
- referring to the 68HC11 data sheet, for fastest data transfer: SPR(1:0) = _____, which will
 clock the data bits at a speed = $f_E / ______ = 1.2288 \text{ MHz} / ______ = ______ \text{ Kbits/sec.}$
- SCK period = duration of one data bit = _____ $\mu\text{Sec.}$ (for scope check)
- duration of two data bits = _____ $\mu\text{Sec.}$ (for scope check)
- duration of four data bits = _____ $\mu\text{Sec.}$ (for scope check)
- duration of transmission of one byte = _____ $\mu\text{Sec.}$
- NOTE: scope measurement from falling edge of first clock pulse, to rising edge of last
 clock pulse = _____ (not quite 8) cycles = _____ $\mu\text{Sec.}$
- initialize SPI by loading SPCR = 0b_____ (best expressed in binary)
- at this stage, interrupts are not necessary: set SPIE = 0
- NOTE: according to 68HC11 data, SPI initialization requires also SPIF to be cleared by
 reading the status register (SPSR)
- as with any communication process, FIRST check if the receiver/listener is ready BEFORE
 sending any data: this is accomplished for the SPI using the SPIF bit
- ALTERNATELY, as suggested in data book: ensure SPIF is cleared by reading from
 the SPSR during initialization. Subsequently, just write to SPDR to start transmission,
 and check on SPIF for completion

IMPORTANT: because Port D is also used for SCI communications via bits 0 and 1, it is important to retain their respective directions when setting the other direction bits for SPI communication, thus a **Bit Set** instruction (**Bset**) is required to set the bits for SPI output pins.

- NOTE: to send 12 bits to the DAC8512: send 2 8-bit bytes, MSBit first, in 2 separate transfers; the DAC8512 shift register simply retains the last 12 bits:



- the above information suggests the following elementary DAC8512 “interface” subroutines ...

4. 68HC11 SPI/DAC PROGRAMMING: SUBROUTINES & TEST

For organized and effective programming, DAC related subroutines can be collected in a separate library file, ex. SPI_Lib.Asm, including:

- b) SPI_Init - initialize SPI to interface with DAC8512
- c) SPI_Data - to send 16-bit value in AccD to DAC via SPI
- d) SPI_Test - generates repeated test pattern out of SPI; quits on Port A bit 0 low
- e) DAC_Ramp - generates simple, periodic test ramp voltage out of DAC; quits on PA0 low
- DAC_Sine - generates simple, periodic test sine voltage out of DAC; quits on PA0 low

→ subroutine **SPI_init**: initialize the SPI

- i. set Port D SPI pin directions (SS, Sck, MOSI = Port D pins ____, ____, ____ = OUT)
- ii. initialize the DAC LD pin high: Port D pin ____
- iii. set up SPCR: enable SPI, Master, set serial data speed, and clock polarity and phase
- iv. read in the status register to clear the SPIF bit
- v. NOTE: to speed up subsequent SPI data transfer subroutine, this exits with X = Ctrl_regs (0x1000)

```
; Useful named constants; the following should be added in Equ_HC11.Inc:
Ctrl_regs    =    0x1000
SPCR         =    0x0028          ; Relative offset from Ctrl_regs
SPSR         =    0x_____
SPDR         =    0x_____
SPIF_bit     =    0b_____
SS_bit       =    0b_____
PortD        =    0x_____
DDRD         =    0x_____

;-----
; SPI_Init: initializes 68HC11 SPI to interface with the DAC8512:
;           SS, SCK, and MOSI = OUT, SCK idle high, rising in the middle of
;           data bit
; INPUT    : none
; OUTPUT   : X = Ctrl_regs (0x1000)
; NOTE     : Port D dir. register (DDRD) ONLY bits 5,4,3 (for ~SS, Sclk,
;           and MOSI) are set (=1) for output. Other bits left un-changed.
;-----
SPI_init:    LdX    #Ctrl_regs

             Bset   _____, X, #0b_____          ; DDRD
             _____, X, _____          ; Start DAC LD high

             LdaA  #0b_____          ; Enable SPI, MSTR=CPOL=CPHA = 1
             _____, X              ; to SPI control reg.

             LdaA  _____, X          ; Clear SPIF: read SPI status

             RTS                    ; SPI_init done
```

→ subroutine **DAC_Data** : 12-bit data output to DAC

- i. send out MSByte first
- ii. wait for SPI to complete transmission: check on the SPIF bit
- iii. send out LSByte next, and again wait for SPI to complete transmission
- iv. finally, pulse DAC ~LD pin low then high

```

;-----
; DAC_data      : sends AccD to DAC8512 via SPI
; INPUT         : - AccD bits 11-0 = data to be sent to DAC8512 via SPI
;               : Note: bits 15-12 irrelevant
;               : - X = Ctrl_regs; required to minimize execution time
; OUTPUT        : none
; EXECUTION TIME: approx. _____ uSec. NOTE: this accounts for:
;               : - StaA and StaB instructions: 2 x ___ E = _____ μSec.
;               : + wait for each byte to be sent out = _____ μSec.
;               : (completion of transmission indicated by SPIF)
;               : NOTE: the CPU waits for SPIF by checking on it using the BrClr
;               : instruction: ___ E, and it will thus execute ___ times
;               : before it finds SPIF = 1
;               : → Time of BrClr = ___ times x _____ μSec. = _____ μSec.
;               : → one each after StaA and StaB: _____ μSec.
;               : + Bclr + Bset + Rts
;-----
SPI_Data:      ; another name, for backward compatibility
DAC_Data:     StaA _____, X                ; 4E. Send MSByte to DAC
              BrClr _____, X, #_____, .      ; ___ uS. Wait for SPIF
              _____, X                    ; ___E. Send LSByte to DAC
              _____, X, #_____, .          ; ___ uS. Wait for SPIF
              Bclr _____, X, #SS_bit        ; ___E. Pulse DAC LD low
              _____, X, #_____          ; ___E.
              Rts                               ; ___E. DAC_data

```

→ SPI_Data total execution time: ≈ _____ μSec.

→ total execution time including Jsr instruction: ≈ _____ μSec.

!! NOTE: the above code waits on the SPIF bit using the BrClr instruction, which takes 7 E cycles to execute. Thus, if the SPIF bit goes high some time during the execution of this instruction, this results in a bit of wasted time. SO, if we require the fastest possible execution of this subroutine, we could reduce this wasted time by actually synchronizing the code to the SPI without polling, by extra NOP instructions to match the SPI data transmission. Recall: SPI rate = $f_E / 2$ → SPI sends one byte in $8 \text{ bits} \times 2 \text{ E cycles} = 16 \text{ E cycles}$, and one NOP instruction executes in: _____ E cycles
 → require: _____ NOP instructions

→ 1-st test of proper SPI operation: even before wiring the SPI to the DAC8512, it should prove useful to verify it's proper operation with the above subroutines, by generating a fixed repeating pattern on the SPI lines, ex. repeatedly send out $0x530F = 0101001100001111$; this produces a useful patterns of single data bits low, high, then double low and high: easy to measure on the scope. NOTE: the DAC_data sends out two separate 8-bit bytes, resulting in two separate bursts of SCK pulses.

```

;-----
; SPI_Test : repeatedly sends out 0x530F to the SPI
;           scope check MOSI: -----
;           Scope trig on ~SS: -----
;           Low duration:      → ← ____ μSec.
;           Sclk:             -----
;           Falling to rising edge: | ____ μSec. |
; INPUT      : none
; OUTPUT     : none
; NOTE      : this terminates when Port A bit 0 is low
;-----
SPI_Test:      _____ ; Init; returns: X = Ctrl_regs
              LdD          _____ ; Data test pattern
1$:           _____ ; Send it out: Jsr = ____ E
              BrSet       PortA, X, #0b00000001, 1$ ; Quit on PA0 - lo
              RTS

```

→ Expected period of repetition:

- Jsr instruction: ____ E = ____ μSec.
- Execution of SPI_Data subroutine: ____ μSec.
- BrClr instruction: ____ E = ____ μSec. **→ TOTAL EXPECTED = ____ μSec.**

→ Expected patterns: for best results, synchronize scope to the simplest signal: SPI SS falling edge

→ measure each of the other two on scope chan. 2, relative to the above

→ expected delay from SS rising edge to SCK pulses:

= RTS instruction in SPI_Data: ____ E + BrSet in SPI_Test: ____ E + Jsr SPI_data: ____ E
+ StaA instruction in SPI_data: ____ E **→ TOTAL = ____ E = ____ μS.**

