Column #92 December 2002 by Jon Williams:

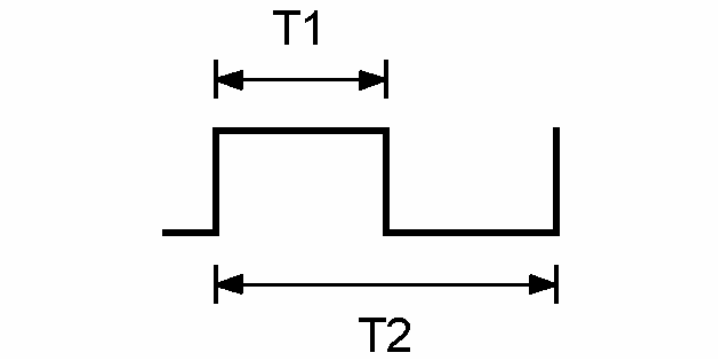
It’s All About Angles

Have I ever told you about my buddy, Chuck? Chuck is a great guy. He's friendly, he's personable and he loves BASIC Stamps. Truth be told, Chuck is actually the grandfather of the BASIC Stamp. That's right, Chuck's son Chip invented the BASIC Stamp way back when. Since then, Chuck has been very involved in Stamps and has been an integral part of BASIC Stamp development since the BS2sx days.

With a background in aerospace, the Chuck-Dawg (my nickname for him since he labels those who please him as "good doggies") enjoys connecting sensors of various types and arrangements to the BASIC Stamp. It's safe to say that he's passionate about sensors; he is always on the lookout for new sensors and how they can be connected to the BASIC Stamp to solve various engineering problems.

On my visit to the California office in October, I was handed a pile of sensors to evaluate and work with – a couple of them we're going to look at here. So let's finish of the year where we started last month, with an angle on sensors; specifically those that give us an indication of angle.

Figure 92.1: Memsic 2125 GW Duty Cycle



$$A\ (g) = (T1 / T2 - 0.5) / 12.5\%$$

Memsic 2125

The first sensor we'll look at is the Memsic 2125 dual-axis accelerometer. The 2125 is identical to the ADXL202, but uses a different technology. Inside, the Memsic 2125 actually has a small heater that warms a "bubble" of air. When the sensor moves, this heat bubble moves and is detected by thermopiles that surround the heater. Changes detected by the thermopiles are conditioned and the outputs are pulses (one for each axis) that correspond to the forces acting on the sensor.

The Memsic 2125 comes in a surface-mount package so Parallax has made a DIP carrier that is convenient for prototyping. In addition to the X and Y pulse outputs, an analog output for die temperature is available [for compensation when needed]. You'll need a separate analog-to-digital converter to measure the temperature output.

Let's keep our focus on acceleration. Take a look at Figure 92.1. This shows the pulse output from the 2125 and the formula for calculating acceleration. For the Memsic 2125GW [that we'll use], the output range is ±2g and the value of T2 is fixed at 10 milliseconds. For 0g, the T1 output will be five milliseconds; a 50% duty-cycle.

As you can see, the math is pretty easy. Here's how we handle it with the BASIC Stamp:

```
Read X Force:
  PULSIN Xin, HiPulse, xRaw
  xRaw = xRaw * 2
  xGForce = ((xRaw / 10) - 500) * 8
  RETURN
```

Since the Memsic 2125 output is a pulse, we'll use PULSIN to measure it. Remember, though, that the value returned by PULSIN is in two-microsecond units. We can adjust for this by multiplying the raw pulse width value by two.

Since our T1 time is in microseconds and there are 1000 microseconds in a millisecond, we need to multiply the T2 value and 0.5 by 1000 to adjust the equation. This makes the math easy for the Stamp and gives us an output with a resolution of 0.001g. Finally, the divide by 12.5% part of the equation is converted to multiplying by eight (1 / 0.125 = 8) – we couldn't ask for a value much more convenient than that.

One of the many useful applications of the 2125 is to measure tilt. At Parallax, we're particularly interested in measuring tilt in robots; like our Toddler and SumoBot. In the Toddler, measuring tilt helps us fine-tune the walking algorithm and ensures that it doesn't ever lean too far to one side or the other. In the SumoBot, we're working on escape algorithms based on measuring robot tilt when the motors have stalled.

Again, we luck out, because within the range useful tilt angles for our projects, the calculation of tilt from g-force can be handled with a linear equation.

Tilt = g x k

where k is taken from the following table (provided by Memsic):

inclination range (°arc)	*k* (°arc/g)	maximum error (°arc)
±10	57.50	±0.02
±20	58.16	±0.16
±30	59.04	±0.48
±40	60.47	±1.13
±50	62.35	±2.24

Now, we could take the easy route and use 62.35 as our conversion factor, but this would give us an error at low tilt angles that we really don't need to tolerate. What we'll do, then, is work backward from this data and determine the output from the 2125 at the angles specified in the chart. Then we can use LOOKDOWN to compare the 2125 output to the chart and a LOOKUP table to determine the proper tilt conversion factor.

Calculating the maximum g-force output for a given range is done by dividing the arc for a range by its conversion factor. This will give us the maximum g-force output for that range.

10 ÷ 57.50 = 0.17391
20 ÷ 58.16 = 0.34387
30 ÷ 59.05 = 0.50804
40 ÷ 60.47 = 0.66148
50 ÷ 62.35 = 0.80192

Remember that the g-force output from the BASIC Stamp code is in 0.001g units, so we'll multiply the results above by 1000 for use in a LOOKDOWN table. Let's look at the code that converts g-force to tilt.

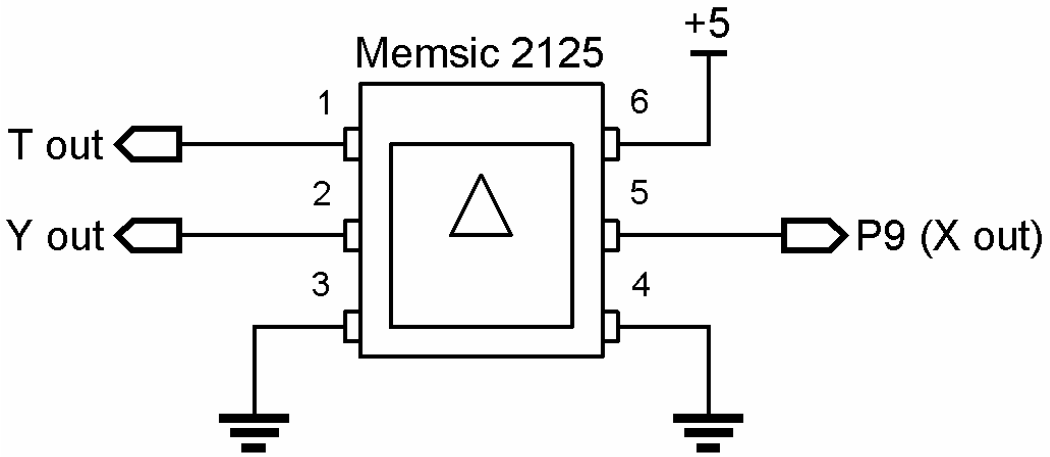
```
Read X Tilt:
  GOSUB Read X Force

  LOOKDOWN ABS xGForce, <=[174, 344, 508, 661, 2000], idx
  LOOKUP idx, [57, 58, 59, 60, 62], mult
  LOOKUP idx, [32768, 10486, 3277, 30802, 22938], frac

  xTilt = mult * (ABS xGForce / 10) + (frac ** (ABS xGForce / 10))

Check SignX:
  IF (xGForce.Bit15 = 0) THEN XT Exit
  xTilt = -xTilt

XT_Exit:
  RETURN
```

The routine starts by reading the g-force with the code we developed earlier. The absolute value of the g-force is compared to LOOKDOWN table entries to determine the index position (in the other tables) of our conversion factor.

Again, we're using the conditional operator with the LOOKDOWN table. In our case, LOOKDOWN will identify the first location of table data that is less than or equal to (<=) the g-force value. This position is moved into the variable idx and serves as an index for the next two

LOOKUP tables. Note that the final value in the LOOKDOWN table is 2000, not 802 as you might expect. This is the maximum output from the Memsic 2125 and assigns the 62.35

Figure 92.2: Memsic 2125 to BASIC Stamp Connection



conversion factor to all tilt angles above 40 degrees. Remember that it's only accurate to ±50 degrees.

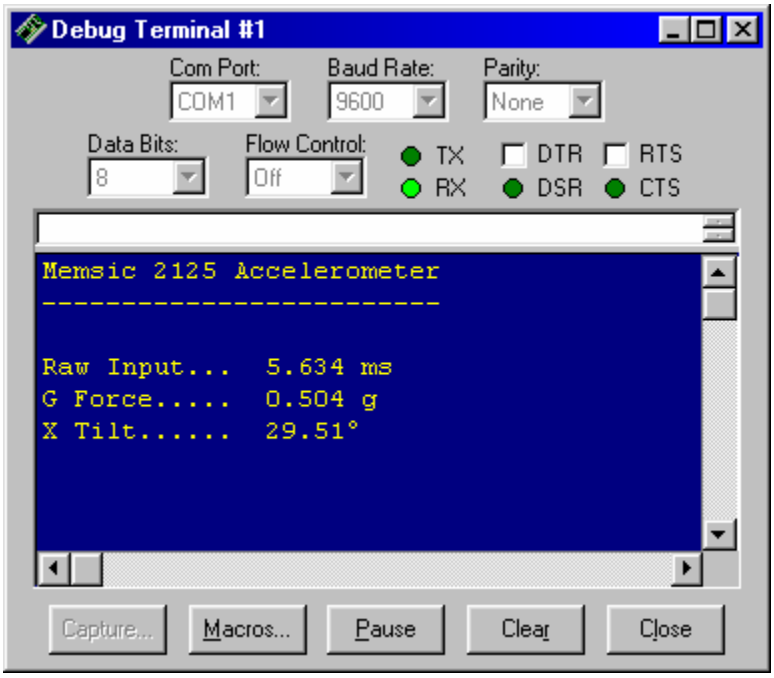
In order to use the ** operator – which gives us the best resolution when multiplying by fractional values – we have to separate the conversion factors into their whole and fractional elements. The first table contains the whole part, the second table contains the fractional part. The entries for the second table are calculated by multiplying the fractional portion of each conversion factor by 65,536.

All that's left to do now is the math. The g-force value is divided by 10 in each part of the equation to prevent a roll-over error, and the ABS (absolute) function must be used because we can't do a division on a negative number. We do a straight multiply with the whole part, use ** with the fractional part, then add them together. The result with be in 0.01 degree units.

In order to know which direction we're tilted, we have to put the sign in. We can do this by checking bit 15 of the g-force value. If this is one, the g-force and tilt is negative.

Connect the Memsic 2125 (Parallax module) as in Figure 92.2 and download the rest of the demo code. If you tilt the board so that the end indicated by the pointer is higher, you'll get an output as in Figure 92.3.

Figure 92.3: Memsic 2125 Sample Output in BASIC Stamp DEBUG Window



Austria Microsystems AS5020-E

The AS5020 is a really neat little sensor; it's an absolute angular position encoder – and a gift from the Chuck-Dawg, no less.

Inside the AS5020 is a Hall effect sensor array, allowing it to determine the angle of a magnetic field that is placed in close proximity to its body. The output is a six-bit value which will give us one of 64 positions, or an angular resolution of 5.6 degrees.

Connecting to the AS5020 is very much like connecting to a DS1620, DS1302, AC0831 or any other 3-wire device. The Chip Select (CS\) pin is brought low to activate the AS5020, then SHIFTIN is used to read the position using the Clock and Data lines. There is just one caveat: the AS5020 requires an extra pulse on the clock line to initiate the measurement cycle. Take a look at Figure 4 to see the AS5020 signaling.

The measurement cycle actually starts on the falling edge of the clock line after the Chip Select is taken low. Don't be fooled into thinking that you can just add an extra bit to the SHIFTIN function, this won't work. The reason is that SHIFTIN expects a data bit to be ready after each clock and in this case that won't happen.

What we'll do to initiate the measurement cycle is use PULSOUT to create that extra transition. As I just mentioned, a high-to-low transition on the clock line is what actually initiates the measurement cycle. After the measurement we use SHIFTIN to collect the position value.

Here's the code:

```
Get Position:
  LOW CSpin
  PULSOUT Cpin, 1
  PAUSE 0
  SHIFTIN Dpin, Cpin, MSBPOST, [position\6]
  HIGH CSpin
  RETURN
```

As you can see, the code follows the communication diagram step-by-step. After selecting the device by bringing the CS\ line low, the measurement pulse is created with PULSOUT. I threw a PAUSE 0 into the code for faster BASIC Stamps, but the routine works fine without it on a stock BS2. The data output from the AS5020 is six bits wide, appearing MSB first and after the clock line falls. The correct SHIFTIN parameter, then, is MSBPOST (sample bit post clock). Since the position value is only six bits, we need to specify that in the SHIFTIN function. If we leave the \6 parameter out, SHIFTIN would collect eight bits and the extra bits would have to be removed from the position value by dividing it by four. After we have the position, the AS5020 is deselected by taking the CS\ line high.

Simple, huh? Okay, what do we do with the data now? Well, that depends on your project. You can download a couple of interesting application notes from Austria Microsystems that show how to use the device for both angular and linear position indication.

We can also convert the position value to an angle with a bit of simple math. The position to degrees conversion factor is calculated by dividing 360 degrees by 64 positions; this gives us 5.625 degrees per position. If we multiply this by 10, we can convert the AS5020 output to tenths degrees with this line of code:

```
degrees = position */ 14400
```

For review, the */ parameter is calculated by multiplying 56.25 by 256.

Figure 92.4: AS5020-E Timing Diagram

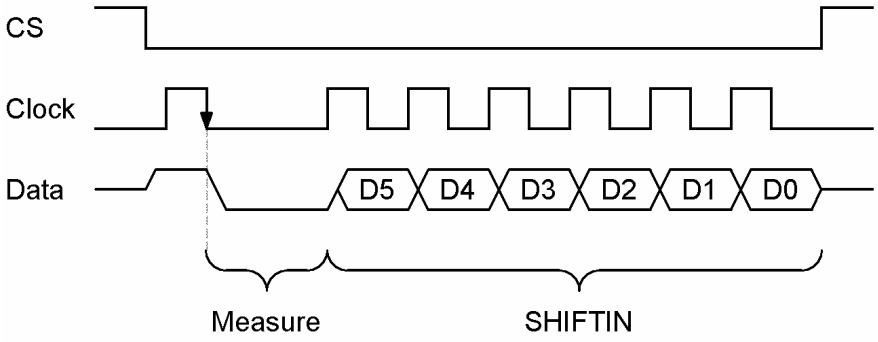
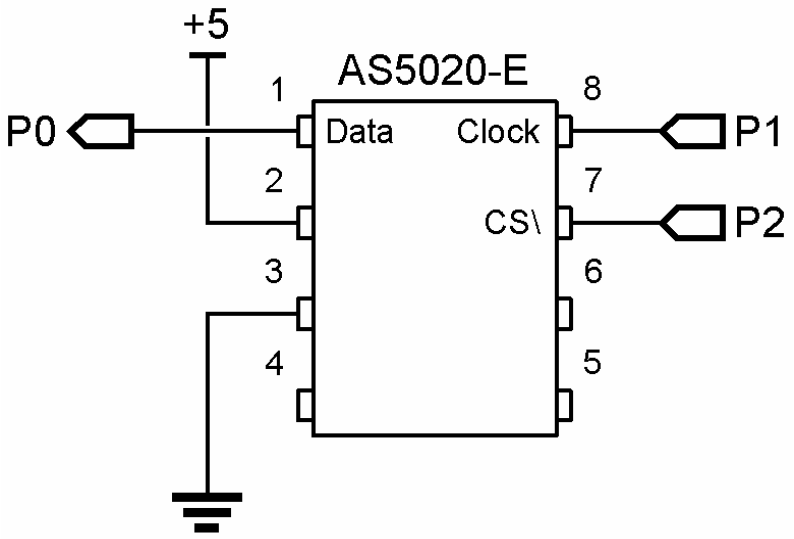


Figure 92.5: AS5020-E to BASIC Stamp Connection



Another feature of the AS5020 is the ability to reprogram its output so that you can align zero to any position you choose. This can be tricky though, because it requires a higher voltage on the programming pin and sending the position data. We don't have to go through that much trouble; we can easily calibrate a project using PBASIC.

The way we'll do it is to take an initial reading, then subtract this value this from 64. This will give us a position offset. Set the project to its "zero" position, then run this code:

```
Set_Offset:
  offset = 0
  GOSUB Get_Position
  offset = 64 - position
```

Now, by adding the offset to the current position, then using the modulus operator, we can adjust the output for our calibrated "zero" position. Add this line off code to the end of the Get_Position subroutine to integrate the offset:

```
position = position + offset // 64
```

Remember, the modulus operator returns the remainder of a division, so the line of code above will keep the position value between 0 and 63, which corresponds to the standard output from the sensor. The difference now is that we've set the zero position and we can reset it as required.

Happy Holidays

Wow, it's hard to believe that another year has come and [nearly] gone – time sure does fly by quickly when you're having fun, doesn't it? I appreciate all of you who have sent e-mail or called with suggestions for the column. I sincerely enjoy hearing from you and about your projects – keep that e-mail coming!

Next year will be very exciting, I'm sure. A big improvement is coming in the BASIC Stamp editor software and the long-awaited next-generation BASIC Stamp should probably show up in 2003 as well. Like the original BASIC Stamp, it's going to change perceptions of what an easy-to-program embedded microcontroller can do.

Finally ... on behalf of my family and my extended family at Parallax, please allow me to wish you and yours a very happy and peaceful holiday season. Let us hope that the new year brings peace and prosperity for every person, no matter their race, creed or religion.

God bless you all.

Memsic 2125
www.memsic.com

Austria Microsystems AS5020-E
www.austriamicrosystems.com

```
' =============================================================================
'
'   Program Listing 92.1
'   File...... MEMSIC2125.BS2
'   Purpose... Memsic 2125 Accelerometer Demo
'   Author.... Jon Williams
'   E-mail.... jwilliams@parallaxinc.com
'   Started... 07 OCT 2002
'   Updated... 08 OCT 2002
'
'   {$STAMP BS2}
'
' =============================================================================


' -----------------------------------------------------------------------------
' Program Description
' -----------------------------------------------------------------------------
'
' Read the pulse output from a Memsic 2125 accelerometer and converts to G-
' force and tilt angle.
'
' Note that the [original] current Memsic 2125 data sheet has an error in the
' G force calcuation.  The correct formula is:
'
' G = ((t1 / 10 ms) - 0.5) / 12.5%


' -----------------------------------------------------------------------------
' Revision History
' -----------------------------------------------------------------------------


' -----------------------------------------------------------------------------
' I/O Definitions
' -----------------------------------------------------------------------------

Xin             CON     9                       ' X input from Memsic 2125


' -----------------------------------------------------------------------------
' Constants
' -----------------------------------------------------------------------------

HiPulse         CON     1                       ' measure high-going pulse
LoPulse         CON     0

MoveTo          CON     2                       ' move to x/y in DEBUG window
ClrRt           CON     11                      ' clear DEBUG line to right
DegSym          CON     176                     ' degrees symbol
```

```
' -----------------------------------------------------------------------------
' Variables
' -----------------------------------------------------------------------------

xRaw            VAR     Word                    ' pulse width from Memsic 2125
xGForce         VAR     Word                    ' x axis g force (1000ths)
xTilt           VAR     Word                    ' x axis tilt (100ths)

idx             VAR     Nib                     ' table index
mult            VAR     Word                    ' multiplier - whole part
frac            VAR     Word                    ' multiplier - fractional part


' -----------------------------------------------------------------------------
' EEPROM Data
' -----------------------------------------------------------------------------


' -----------------------------------------------------------------------------
' Initialization
' -----------------------------------------------------------------------------

Setup:
  PAUSE 250                                     ' let DEBUG window open
  DEBUG "Memsic 2125 Accelerometer", CR
  DEBUG "-------------------------"


' -----------------------------------------------------------------------------
' Program Code
' -----------------------------------------------------------------------------

Main:
  GOSUB Read X Tilt                             ' reads G-force and Tilt

Display:
  DEBUG MoveTo, 0, 3
  DEBUG "Raw Input...  ", DEC (xRaw / 1000), ".", DEC3 xRaw, " ms", ClrRt, CR

  DEBUG "G Force..... ", (xGForce.Bit15 * 13 + " ")
  DEBUG DEC (ABS xGForce / 1000), ".", DEC3 (ABS xGForce), " g", ClrRt, CR

  DEBUG "X Tilt...... ", (xGForce.Bit15 * 13 + " ")
  DEBUG DEC (ABS xTilt / 100), ".", DEC2 (ABS xTilt), DegSym, ClrRt

  PAUSE 200
  GOTO Main


' -----------------------------------------------------------------------------
```

```
' Subroutines
' -----------------------------------------------------------------------------

Read X Force:
  PULSIN Xin, HiPulse, xRaw                     ' read pulse output
  xRaw = xRaw * 2                               ' convert to microseconds

  ' g = ((t1 / 0.01) - 0.5) / 12.5%            ' correction from data sheet
  '
  xGForce = ((xRaw / 10) - 500) * 8             ' convert to 1/1000 g
  RETURN


Read X Tilt:
  GOSUB Read X Force

  ' tilt = g x k
  '
  ' Select tilt conversion factor based on static
  ' G force.  Table data derived from Memsic specs.

  LOOKDOWN ABS xGForce, <=[174, 344, 508, 661, 2000], idx
  LOOKUP idx, [57, 58, 59, 60, 62], mult
  LOOKUP idx, [32768, 10486, 2621, 30802, 22938], frac

  ' G Force is divided by 10 to prevent roll-over errors at end
  ' of range.  Tilt is returned in 100ths degrees.

  xTilt = mult * (ABS xGForce / 10) + (frac ** (ABS xGForce / 10))

Check_SignX:
  IF (xGForce.Bit15 = 0) THEN XT Exit           ' if positive, skip
  xTilt = -xTilt                                ' correct for g force sign

XT Exit:
  RETURN
```

```
' =============================================================================
'
'   Program Listing 92.2
'   File...... AS5020.BS2
'   Purpose... Austria Microsystems AS5020-E interface
'   Author.... Jon Williams
'   E-mail.... jwilliams@parallax.com
'   Started... 18 OCT 2002
'   Updated... 18 OCT 2002
'
'   {$STAMP BS2}
'
' =============================================================================


' -----------------------------------------------------------------------------
' Program Description
' -----------------------------------------------------------------------------
'
' This program reads and displays the position data from an AS5020 absolute
' angular position encoder.
'
' Data for the AS5020-E is available from www.austriamicrosystems.com


' -----------------------------------------------------------------------------
' Revision History
' -----------------------------------------------------------------------------


' -----------------------------------------------------------------------------
' I/O Definitions
' -----------------------------------------------------------------------------

Dpin            CON     0                       ' data in from AS5020
Cpin            CON     1                       ' clock out to AS5020
CSpin           CON     2                       ' chip select (active low)


' -----------------------------------------------------------------------------
' Constants
' -----------------------------------------------------------------------------

MoveTo          CON     2                       ' move to x/y in DEBUG window
ClrRt           CON     11                      ' clear DEBUG line to right
DegSym          CON     176                     ' degrees symbol


' -----------------------------------------------------------------------------
' Variables
```

```
' -----------------------------------------------------------------------------

position        VAR     Byte                    ' angular position from AS5020
degrees         VAR     Word                    ' convert to degrees
offset          VAR     Byte                    ' position offset


' -----------------------------------------------------------------------------
' EEPROM Data
' -----------------------------------------------------------------------------


' -----------------------------------------------------------------------------
' Initialization
' -----------------------------------------------------------------------------

Initialize:
  HIGH CSpin                                    ' deselect AS5020
  LOW Cpin

  PAUSE 250                                     ' let DEBUG open
  DEBUG CLS
  DEBUG "AS5020 Angular Position Sensor", CR
  DEBUG "-----------------------------"


' -----------------------------------------------------------------------------
' Program Code
' -----------------------------------------------------------------------------

Main:
  GOSUB Get Position                            ' read position from AS5020

  ' degrees (tenths) = position * 56.25
  '
  degrees = position */ 14400

  DEBUG MoveTo, 0, 3
  DEBUG "Position... ", DEC position, ClrRt, CR
  DEBUG "Angle..... ", DEC (degrees / 10), ".", DEC1 degrees, degSym, ClrRt

  GOTO Main
  END


' -----------------------------------------------------------------------------
' Subroutines
' -----------------------------------------------------------------------------

Get Position:
  LOW CSpin                                     ' select device
```

```
  PULSOUT Cpin, 1                                ' start measurement
  PAUSE 0                                        ' allow measurement
  SHIFTIN Dpin, Cpin, MSBPOST, [position\6]      ' load data
  HIGH CSpin                                     ' deselect device
  position = position + offset // 64             ' account for offset
  RETURN


Set Offset:
  offset = 0                                     ' clear offset
  GOSUB Get Position                             ' get current position
  offset = 64 - position                         ' calculate new "0" offset
  RETURN
```