Freescale Semiconductor

Order this document
by AN1222/D

AN1222

Arithmetic Waveform Synthesis with the HC05/08 MCUs

by Mark McQuilken & Mark Glenewinkel
CSIC Applications

INTRODUCTION

IThis application note is intended to demonstrate the use of Arithmetic Synthesis to create sinusoidal waveforms from a microcontroller unit (MCU). Given an accumulation constant predefined in memory, a very precise sinusoidal waveform can be produced from a table of sinusoidal values in memory. The values selected from the table are then sent out an MCU port to a digital-to-analog converter (DAC). This application note has been written for the HC08 MCU. Although cycle execution time will be different, the program listing for the HC08 is also applicable to the HC05.

BACKGROUND

The process of producing tones using an accumulated value that is used to point to the next output time (magnitude) sample is called "Arithmetic Synthesis". This is contrasted to Direct Digital Synthesis where the "distance" between each sample taken from sinewave table is constant.

Arithmetic Synthesis utilizes a standard DAC look-up, phase value table consisting of (in this case) 256 phase values for a single cycle of a sine wave. The position in this table is determined by two bytes of data: the MSB which is the integer index into the DAC look-up table, and the LSB which is the fractional depth into the table. Since this fractional portion is not used to directly address a sine sample directly, the effect is evident only when the continual accumulation of this fraction causes an overflow into the integer portion. This effect may be seen in a simple example of repeatedly adding (accumulating) a fixed value while only attributing significance to the integer portion:

Accumulation (Truncated Integer)		**Addend (Integer Fraction)**	**Result (Integer Fraction)**
0	+	1.25	1.25
1	+	1.25	2.50
2	+	1.25	3.75
3	+	1.25	5.00
5	+	1.25	6.25
6	etc ...		

The effect of the fractional accumulation on the integer portion is an occasionally "skipped" value (like the value of 4 that was skipped above). The integer part of the accumulator is then used to point into the sine table to obtain a magnitude for that time sample. The frequency of occurance of this skipping is a function of the fractional value. Hence, we can determine, with the appropriate choice of fraction and integer, the output frequency of the digitized sinewave to a high degree of accuracy. The exact mathematical relationship, for a 256 phase value table, is:

Accumulation Constant = 256 x Desired Frequency x Sampling Period

The sampling period would be:

Sampling Period = E clock period * # of cycles in loop

Once the accumulation constant, D, is determined, it must be put into the memory locations Int_K and Frac_K. This calculation is shown below:

D = 6.575
Int_K = 6 = $06
Frac_K = 0.575 * 256 = 147.2
= $93

Once the accumulation constant is defined, the synthesizer is ready to be used. This application note uses the HC08 MCU but the algorithm can be used with any MCU so long as the algorithm is followed and the accumulation constant is defined for the appropriate sampling period.

Arithmetic synthesis produces some artifacts that are not desirable in some applications. One of those artifacts is called "phase noise" (or phase jitter). With an ideal sinewave, the period is fixed and unvarying for every cycle. This means that the instantaneous frequency and the average frequency are the same. What occurs with an arithmetically synthesized waveform, however, is that the instantaneous frequency changes from cycle-to-cycle due to the "sample skipping" performed in the AS algorithm while the average frequency remains quite stable and precise. This cycle-to-cycle variation in instantaneous frequency is called phase noise. Applications which are sensitive to changing instantaneous frequency and/or to the additional spectral components produced by the jitter (these components would be categorized anywhere from distortion to just plain noise, depending on the system) may not want to use AS as the primary method of signal synthesis.

TESTING OF THE ARITHMETIC SYNTHESIZER

In order to test the arithmetic synthesizer, any 8 bit parallel DAC can be used. An Analog Devices AD557JN was used to test our code. The 557 is an easy to use DAC. A basic schematic for the 557 is listed below in Figure #1. The 8 bit digital waveform data is sent to the DAC and the conversion occurs immediately after receiving the information. The sampling period of the waveform is determined by the speed at which data is written to the DAC port. If other MCUs are to be used with this basic circuit, make sure the sampling frequency does exceed the specifed output settling time. Please refer to the AD557 data sheet if more information is needed.

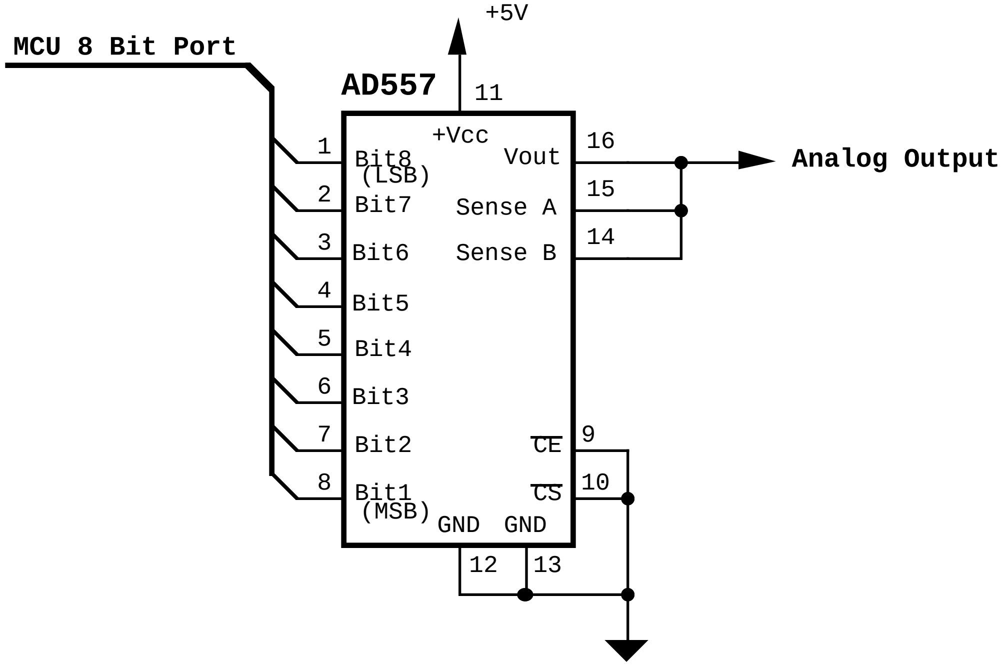


Figure #1 - Basic schematic for the AD557 Digital-to-Analog Converter

DESCRIPTION OF THE ARITHMETIC SYNTHESIS SOFTWARE

The flowchart and code listing written to illustrate arithmetic synthesis is given at the end of this application note. The file name is called ARSYN8.ASM. The code written to execute the loop routine takes 29 cycles. Assuming the HC08 is running at a speed of 8 MHz, the sampling period would be:

$$\begin{aligned} \text{Sampling period} &= \text{E clock period} * 29 \text{ cycles} \\ &= 125\text{nsec} * 29 \text{ cycles} \\ &= 3.625 \text{ usec} \end{aligned}$$

Let's say that you want to produce a sinewave with a frequency of 8 kHz. The accumulation constant will be:

$$\begin{aligned} D &= 256 * 8000\text{Hz} * 3.625\text{usec} \\ &= 7.424 \end{aligned}$$

The accumulation constant must now be put into the memory locations Int_K and Frac_K. These numbers are shown below:

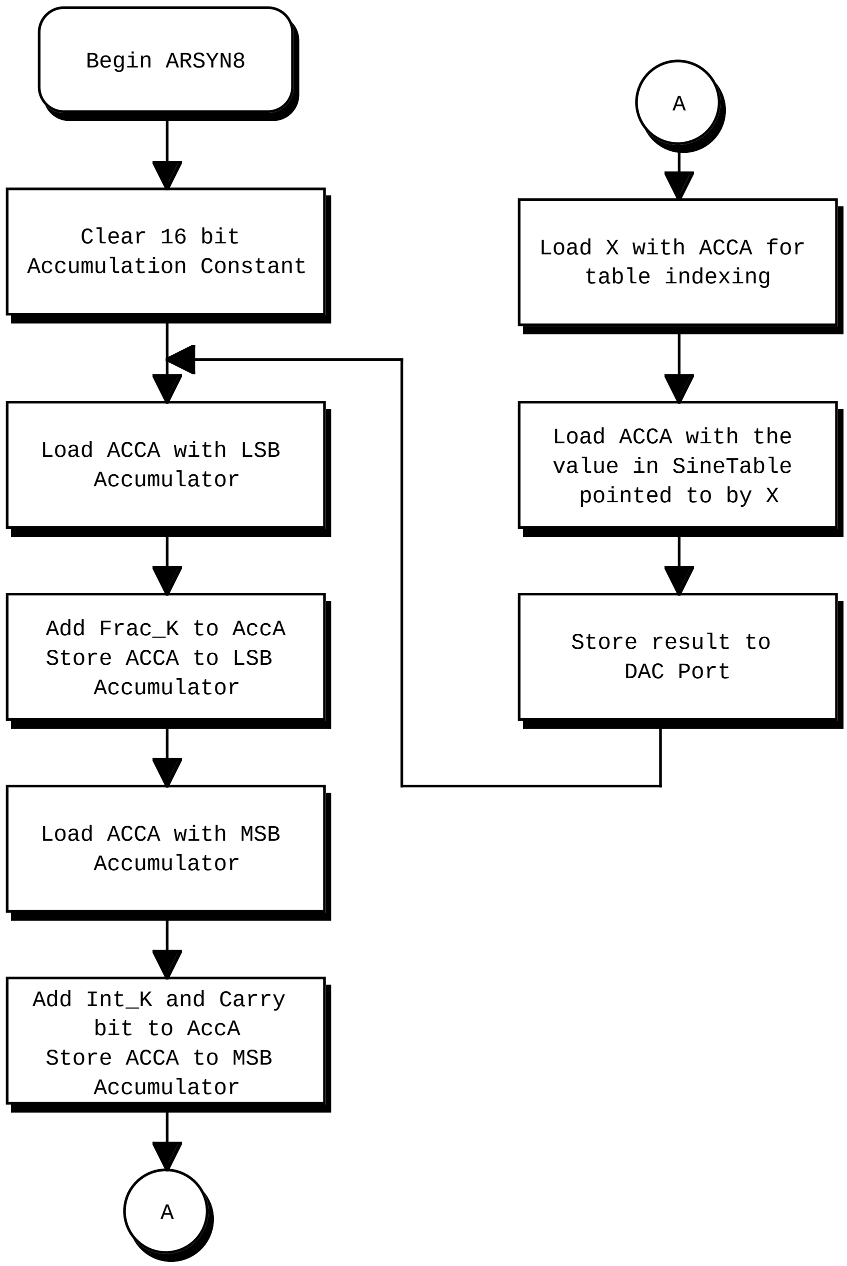
$$\begin{aligned} \text{Int\_K} &= \$07 \\ \text{Frac\_K} &= 0.424 * 256 = 108.544 \\ &= \$6C \end{aligned}$$

The routine is written using an infinite loop. This is not the most practical application of the algorithm but it allows experimentation and measurement of the waveform. Some applications may need a waveform to last for a specified length of time. This can be done by adding a counter into the loop. The sampling period will be affected so the accumulation constant must be changed to reflect the new sampling period.

The sampling period will change when you use an HC05 MCU. Be sure to recalculate the sampling period with the HC05's bus frequency and cycle counts. The code listed will assemble with an HC05 assembler.

ARSYN8.ASM and can be downloaded from the Freescale MCU Bulletin Board Service. The BBS number is (512) 891-3733. The serial protocol is 1200 or 2400, 8 bits, 1 stop bit, and no parity. The file is located on the CSIC bulletin board in the APPNOTES directory.

ARSYN8 - ARITHMETIC SYNTHESIS FLOWCHART



ARSYN8.ASM - ARITHMETIC SYNTHESIS CODE LISTING

```
**********************************************************************
*
* Program Name: ARSYN8.ASM ( Arithmetic Synthesizer )
* Revision: 1.00
* Date: February 3,1993
*
* Written By: Mark Glenewinkel
*              CSIC Applications
*
* Assembled Under: P&E Microcomputer Systems IASM08
*
*       ********************************
*       *       Revision History       *
*       ********************************
*
*       Rev     0.50    12/15/93        M.A. McQuilken
*                       HC05 version to be translated to HC08 code
*
*       Rev     0.60    01/21/93        M.R. Glenewinkel
*                       Added more comments
*
*       Rev     1.00    02/18/93        M.R. Glenewinkel
*                       HC08 version
*
**********************************************************************
*
* Program Description:
*
*       This routine produces a sinusoid of a specified frequency at
*       the output of a Digital-to-Analog Converter (DAC) attached
*       to Port X (set by the user) of an HC08.
*
*       Basically, this method utilizes a standard DAC look-up table
*       consisting of (in this case) 256 phase values for a single
*       cycle of a sinewave. The position in this table is determined
*       by two bytes of data: the MSB which is the integer index into
*       the DAC look-up table, and the LSB which is the fractional
*       depth into the table. Since this fractional portion is not
*       used to directly address a sine sample directly, its effect
*       is made only when the continual accumulation of this fraction
*       causes an overflow into the integer portion.
*
*       We can determine, with the appropriate choice of fraction
*       and integer, the output frequency of the digitized sinewave
*       to a high degree of accuracy. The exact mathematical
*       relationship, for 256 phase values, is:
*
*       Accumulation Constant = 256 x Desired Frequency x Sample Time
*
*       The sample time, in this case, is 3.625 usec using an HC08
*       running with an 8MHz E clock. The sample period is calculated
*       by determining the number of instructions within the
*       generating loop (29 cyc) and multiplying the number by the
*       bus clock period. In this example an 8kHz sinewave is to be
*       sythesized. The accumulation constant will be:
```

```
*
*                              D = 7.424
*
*       Thus, in the source code, the memory locations Int_K and
*       Frac_K should contain $07 and $6C, respectively, upon entering
*       the ArithSyn routine.
*
********************************* CAUTION ****************************
* Please understand the impact that code and/or hardware changes may
* have on the desired, synthesized output frequency. Before making
* changes be certain that you understand the ramifications.
**********************************************************************
*
********************************* NOTE *******************************
* This code assumes that you've got an appropriate DAC on PortB (in
* this case, the testbed consisted of an AD557JN on PortB) and that
* you've made PortB an output port in your initialization code.
**********************************************************************
*
* TASK DATA:
*  Input Variables     Output Variables        Description
*  ----------------    -----------------       -------------
*  Frac_K                                      Enter with the
*                                               appropriate
*                                               fractional
*                                               accumulate value.
*  Int_K                                       Enter with appropriate
*                                               integer accumulation
*                                               value.
*
*
*
* LOCAL DATA:
*  Input Variables     Output Variables        Description
*  ----------------    -----------------       -------------
*  Cntr                Cntr                    Determines length of
*                                               time the sinewave is
*                                               generated with a
*                                               maximum value of
*                                               approx. 4.5 msec.
*  AccumLSB            AccumLSB                Least significant
*                                               byte of 16-bit
*                                               phase accumulator.
*  AccumMSB            AccumMSB                Most significant
*                                               byte of 16-bit
*                                               phase accumulator.
*                                               This is the value
*                                               that is used to point
*                                               into the sine table.
*  ACCA                ACCA                    Misc. computational
*                                               use.
*  X                   X                       Misc. computational
*                                               use.
*
**********************************************************************
*
* Register and Variable Equates
*
```

```
PortB           equ     $01
DAC             equ     PortB
*
**********************************************************************
*
* Memory
*
                ORG     $50
AccumLSB        RMB     1
AccumMSB        RMB     1
Frac_K          RMB     1
Int_K           RMB     1
*
**********************************************************************

                ORG     $6E00           ;beginning of program area
START           EQU     *

**********************************************************************

* Main Routine

* We must prepare the workspace, any nonzero variables could
* alter our process.

ArithSyn        clr     AccumLSB
                clr     AccumMSB


* At this point we're ready to actually begin the process of
* generating the sinewave. The accumulation is done with an 8-bit
* fraction and an 8-bit integer.
* So, basically, we're going to have the HC08 do a 16-bit addition:

SignalGen       lda     AccumLSB        ;3 - Get current LSB value
                                        ;  of the phase accum.
                add     Frac_K          ;3 - The fractional
                                        ;  constant is added to
                                        ;  the LSB of the phase
                                        ;  accumulator.
                sta     AccumLSB        ;3 - Make sure that
                                        ;  the updated value is
                                        ;  kept for the next time
                                        ;  through the loop.

* Here's the second half of our 16-bit addition. This will
* propagate any overflow from the 8-bit addition of the
* AccumLSB and Frac_K into AccumMSB:

                lda     AccumMSB        ;3 - Get current MSB value
                                        ;  of the phase accum.
                adc     Int_K           ;3 - The integer constant
                                        ;  is added to the MSB of
                                        ;  the phase accum.
                                        ;    Notice the addition
                                        ;  with carry.
                sta     AccumMSB        ;3 - Save for next time
                                        ;  through the loop.
```

```
* If we've made it here, then we are ready to turn our phase
* value into a sine wave at the output of the DAC:

                tax                     ;1 - ACCA contains the
                                        ;  integer portion of the
                                        ;  16-bit phase value.
                                        ;  We need to move it into
                                        ;  the X-reg to do a table
                                        ;  look-up.
                lda     SineTable,X     ;4 - Get the sine value
                sta     DAC             ;3 - Send it to the DAC
                                        ;  to create a real-world
                                        ;  signal.

                bra     SignalGen       ;5 - Branch to top of
                                        ;  signal generation
                                        ;  for an infinite loop

***********************************************************************

* Tables

SineTable       FCB     $80,$83,$86,$89,$8C,$90,$93,$96
                FCB     $99,$9C,$9F,$A2,$A5,$A8,$AB
                FCB     $AE,$B1,$B3,$B6,$B9,$BC,$BF,$C1
                FCB     $C4,$C7,$C9,$CC,$CE,$D1,$D3
                FCB     $D5,$D8,$DA,$DC,$DE,$E0,$E2,$E4
                FCB     $E6,$E8,$EA,$EB,$ED,$EF,$F0
                FCB     $F1,$F3,$F4,$F5,$F6,$F8,$F9,$FA
                FCB     $FA,$FB,$FC,$FD,$FD,$FE,$FE
                FCB     $FE,$FF,$FF,$FF,$FF,$FF,$FF,$FF
                FCB     $FE,$FE,$FE,$FD,$FD,$FC,$FB
                FCB     $FA,$FA,$F9,$F8,$F6,$F5,$F4,$F3
                FCB     $F1,$F0,$EF,$ED,$EB,$EA,$E8
                FCB     $E6,$E4,$E2,$E0,$DE,$DC,$DA,$D8
                FCB     $D5,$D3,$D1,$CE,$CC,$C9,$C7
                FCB     $C4,$C1,$BF,$BC,$B9,$B6,$B3,$B1
                FCB     $AE,$AB,$A8,$A5,$A2,$9F,$9C
                FCB     $99,$96,$93,$90,$8C,$89,$86,$83
                FCB     $80,$7D,$7A,$77,$74,$70,$6D
                FCB     $6A,$67,$64,$61,$5E,$5B,$58,$55
                FCB     $52,$4F,$4D,$4A,$47,$44,$41
                FCB     $3F,$3C,$39,$37,$34,$32,$2F,$2D
                FCB     $2B,$28,$26,$24,$22,$20,$1E
                FCB     $1C,$1A,$18,$16,$15,$13,$11,$10
                FCB     $0F,$0D,$0C,$0B,$0A,$08,$07
                FCB     $06,$06,$05,$04,$03,$03,$02,$02
                FCB     $02,$01,$01,$01,$01,$01,$01
                FCB     $01,$02,$02,$02,$03,$03,$04,$05
                FCB     $06,$06,$07,$08,$0A,$0B,$0C
                FCB     $0D,$0F,$10,$11,$13,$15,$16,$18
                FCB     $1A,$1C,$1E,$20,$22,$24,$26
                FCB     $28,$2B,$2D,$2F,$32,$34,$37,$39
                FCB     $3C,$3F,$41,$44,$47,$4A,$4D
                FCB     $4F,$52,$55,$58,$5B,$5E,$61,$64
                FCB     $67,$6A,$6D,$70,$74,$77,$7A,$7D
```

```
*************************************************************************

* Vector Setup

                ORG     $FFFE
                DW      START           ;set up reset vector

*************************************************************************
```


How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com