

Edited by Bill Travis and Anne Watson Swager

Self-modifying code extends addressing mode

Paul Sofianos, Motorola Inc, Tempe, AZ

As just about any assembly-language programmer knows, self-modifying code (SMC) is usually undesirable, unintentional, and destructive. However, the SMC routine in **Listing 1** is extremely useful to extend the indexed, 16-bit offset addressing mode of the venerable HC05 μ C from 8 bits, or 256 locations, to the full 13-bit (8-kbyte) memory address space that the μ C's architecture supports. This routine is very useful for error tables, text messages tables, or any array manipulation for a large number of elements. (You can download **Listing 1** from EDN's Web site: www.ednmag.com. At the registered-user area, go into the Software Center to download the file from DI-SIG, #2280.)

Indexed addressing with offset is useful for selecting an element of byte length, L, in an N-element table. In general, you calculate the effective addresses of this element as follows:

$$EA = TA + L3N + EO,$$

where EA=effective address of the memory location; TA=the absolute address of the start of the element table; L=number of bytes associated with each table element; N=the desired element number, beginning with zero; and EO=element offset, which includes all integer values from 0 to L-1.

The HC05 calculates the quantity $L3N+EO$ and places the result in the index register. The μ C then adds this

dynamic value to the static quantity, TA, to arrive at the effective address. Unfortunately, the index register of the

LISTING 1—SELF-MODIFYING-CODE SUBROUTINE

```

EXAMP      EQU      $03      ;Example table element (0 to N) to transfer from the
SIZE       EQU      $0E      ;For this example, all elements have length 14 (base 10),
                                ;TABLE to TARGET

*-----*
*      Ram Table
*-----*
SMCRAM     ORG      $0050
RMB        RMB      $4      ;Self-modifying-code subroutine
TEMPX      RMB      $1      ;Temp IDX storage
TEMPA      RMB      $1      ;Temp ACC storage
TARGET     RMB      SIZE    ;Target location of element length SIZE

*-----*
*      Initialization
*-----*
INIT       ORG      $0400
LDA        LDA      SMCROM    ;Start of rom
STA        STA      SMCROM    ;Initialize the self-modifying-code.
LDA        LDA      SMCROM+$3 ;LDA instruction
STA        STA      SMCROM+$3 ;RTS instruction

*-----*
*      Program start
*-----*
BEGIN      LDX      #EXAMP    ;Transfer the example element
           JSR      XFER      ;Transfer all bytes from the TABLE rom to TARGET ram
           STOP

*-----*
*      Subroutines
*-----*
*      Subroutine XFER
*      This routine transfers an element of the array TABLE(X) to TARGET.
*      The length of each element, in bytes, is SIZE: (1<=SIZE<=255).
*      The number of the element to be transferred is in IDX.
*      Both ACC and IDX are returned intact.
XFER       STX      TEMPX     ;Temporarily save IDX
           STA      TEMPA     ;Temporarily save ACC
           LDA      #SIZE     ;Calculate the effective address of the first
                                ;byte for the desired element.
           MUL
           ADD      SMCROM+$02 ;Add the element offset to the TABLE offset
           STA      SMCROM+$02
           TXA
           ADC      SMCROM+$01
           STA      SMCROM+$01
           CLRX

           ;Begin transferring the individual bytes, starting with the
           ;byte located at the effective address and ending with the byte
           ;located at the effective address + SIZE - 1
GETBYTE    JSR      SMCROM     ;Get the byte by extended addressing
           STA      TARGET,X  ;Save in the TARGET ram
           INC      SMCROM+$02 ;Add 1 (double-precision) to the SMCROM extended address
           BNE      NOINC
           INC      SMCROM+$01

NOINC      INCX
           CPX      #SIZE     ;Update the index counter
           BNE      GETBYTE   ;Quit when SIZE bytes have been transferred
           LDX      TEMPX     ;Restore IDX
           LDA      TEMPA     ;Restore ACC
           RTS

*-----*
*      Subroutine SMCROM (Dummy)
*      This dummy SMCROM (Self-modifying-code ROM) routine is transferred
*      to SMCRAM during program initialization. Extended address TABLE, the starting
*      address of the data table, is used to calculate an absolute address of a byte
*      of data in the table which is then transferred into ACC.
SMCRAM     LDA      TABLE    ;Absolute addressing pointing to the start of the
           RTS              ;message table

*-----*
*      Data table
*-----*
*      This is the actual data table which can contain any number of entries (up to
*      the limits of the HC05's rom space), with all entries of length SIZE.
TABLE      FCB      'TABLE_ENTRY_00'
           FCB      'TABLE_ENTRY_01'
           FCB      'TABLE_ENTRY_02'
           FCB      'TABLE_ENTRY_03';Selected element for this example
           .
           .
           .
           FCB      'TABLE_ENTRY_99'

*-----*
*      Vectors
*-----*
RES        ORG      $1FEE
           FDB      INIT      ;Reset vector
    
```

Self-modifying code extends addressing mode	95
VHDL procedure dynamically opens a file.....	96
Configure buck converter for boost operation.....	98
Circuit eases three-phase monitoring	100
PLL forms simple MSK demodulator	102
μ C makes inexpensive sine-wave generator	104
74ACT74 makes low-skew clock divider	106

HC05 is only 8 bits long, limiting L3N+EO to values of 0 to 255, which is inadequate for large tables.

The SMC routine in **Listing 1** easily overcomes this limitation. Simply put, the routine copies a dummy static-command set that employs extended addressing from ROM to RAM. The code calculates

an effective address for EO=0 and stores the result as an absolute address, such as TA, for this RAM command. The code then executes this RAM command, fetching a single byte from ROM at an absolute, extended address and saving the byte in RAM by using indexed, 8-bit offset addressing. The routine fetches successive

memory locations from the data table and places the contents in the target table in RAM. This process continues until the transfer of all L bytes of the desired element is complete. (DI #2280)

To Vote For This Design,
Circle No. 371

VHDL procedure dynamically opens a file

Jacques Behar, Rockwell Semiconductor Systems, San Diego, CA

A simple VHDL-87 procedure opens a file whose name a command file or user conveys in runtime. You can apply this technique to the reading or writing of data, control, and status files. This procedure is useful for replacing the input stimuli file of a testbench without recompiling the code (with a different stimuli file name) or rename the stimuli file name. In addition, the procedure can write the simulation results to a file whose name consists

of the input file name and any preferred extension.

The example in **Listing 1** shows the skeleton of a program that reads and sums a file of integers. In this example, the program first reads the

LISTING 1—VHDL-87 PROCEDURE

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use std.textio.all;

-----
entity READ_FILE is
end READ_FILE;

-----
architecture JB of READ_FILE is

signal READ_FILE_ENDED : std_logic := '0';
signal FILENAME       : string(1 to 80) := (others => ' ');
signal FILENAME_LEN   : integer;
signal SINT            : integer;
signal READ_NEW_INT   : std_logic := '0';

begin

-----
MAIN: process

file Fname : text is in: "name.cmd"; --script file
variable VLINE : line;
variable VSTR : string(1 to 80) := (others => ' ');
variable VSUM : integer := 0;

begin
readline(Fname, VLINE);
FILENAME_LEN <= VLINE'length;
read(VLINE, VSTR(1 to VLINE'length));
FILENAME <= VSIR; --filename to process

while (READ_FILE_ENDED /= '1') loop
READ_NEW_INT <= not READ_NEW_INT; --get new read
wait for 10 ns; --set the read cycle time
VSUM := VSUM + SINT; --sum numbers read as an example
end loop;
wait;

end process MAIN;

end JB;

-----
-- configuration
-----
configuration CFG_READ_FILE of READ_FILE is
for JB
end for;
end CFG_READ_FILE;

```

LISTING 2—VHDL READ_FILE PROCESS

```

RD_FILE: process
variable VLINE : line;
variable VINT : integer;

procedure READ_FILE_PROCEDURE(fname: string) is
file ifile : text is in fname;
begin
while (NOT endfile(ifile)) loop
readline(ifile, VLINE);
read(VLINE, VINT);
SINT <= VINT;
if (endfile(ifile)) then exit; end if;
wait on READ_NEW_INT;
end loop;
end READ_FILE_PROCEDURE;

begin
wait until READ_NEW_INT = '1';
READ_FILE_PROCEDURE(FILENAME(1 to FILENAME_LEN));
READ_FILE_ENDED <= '1';
end process RD_FILE;

```

input data file name from the command file “name.cmd” in the first four lines after “begin.” Alternatively, the user can interactively enter the data file name. A VHDL procedure embedded in a concurrent VHDL process achieves the dynamic file access. The main process uses the READ_FILE process in this example (**Listing 2**). Upon activation, the READ_FILE process invokes the READ_

FILE_PROCEDURE. The FILENAME signal conveys the name of the read file. Note that the length of the file name is also necessary, and the FILENAME_LEN signal conveys this length to the READ_FILE procedure.

The MAIN process has sole control over each access to the input file, which is obvious in this simple example that reads a new integer each time the MAIN process toggles the READ_NEW_INT signal. In some complex cases, the MAIN process could conditionally open a file after part of the simulation is complete. At the end of the file, the procedure exits automatically, and the file closes.

You can download both listings from EDN’s Web site: www.ednmag.com. At the registered-user area, go into the Software Center to download the file from DI-SIG, #2281. (DI #2281)

To Vote For This Design,
Circle No. 372

Configure buck converter for boost operation

Mehrzad Koohian, Semtech Corp, Newbury Park, CA

Buck converters are inherently different from boost converters, because buck converters typically use the high side of the output as the power switch's reference. However, a buck converter with a floating output drive section is configurable as a boost controller (**Figure 1**). This circuit configures the SC1101 buck controller for a 5 to $\pm 12\text{V}$ boost with $\pm 500\text{ mA}$ of output current. The BST pin, which normally connects to a high-side drive supply in a buck converter, connects to V_{CC} to drive the ground-referenced MOSFET. By tying PGND to circuit ground, the SC1101 becomes a boost controller, yielding 12V from 5V. An output charge-pump voltage inverter provides -12V at 0.5A as well.

The sense resistor, R_1 , serves two purposes. First, it assures proper start at power-up with full load by limiting the duty cycle. With the output capacitors not charged, a full-load condition demands

high peak inductor currents. If the duty cycle exceeds a maximum limit, the inductor does not have a chance to discharge and will saturate. By limiting the peak currents and thus the duty cycle, the output capacitors can charge in several cycles upon start-up, thus preventing inductor saturation. R_1 also limits switch current during an overload or short circuit. In this application, a value of 0.012V provides for peak-current limiting and allows the circuit to deliver the required output current of $\pm 500\text{ mA}$.

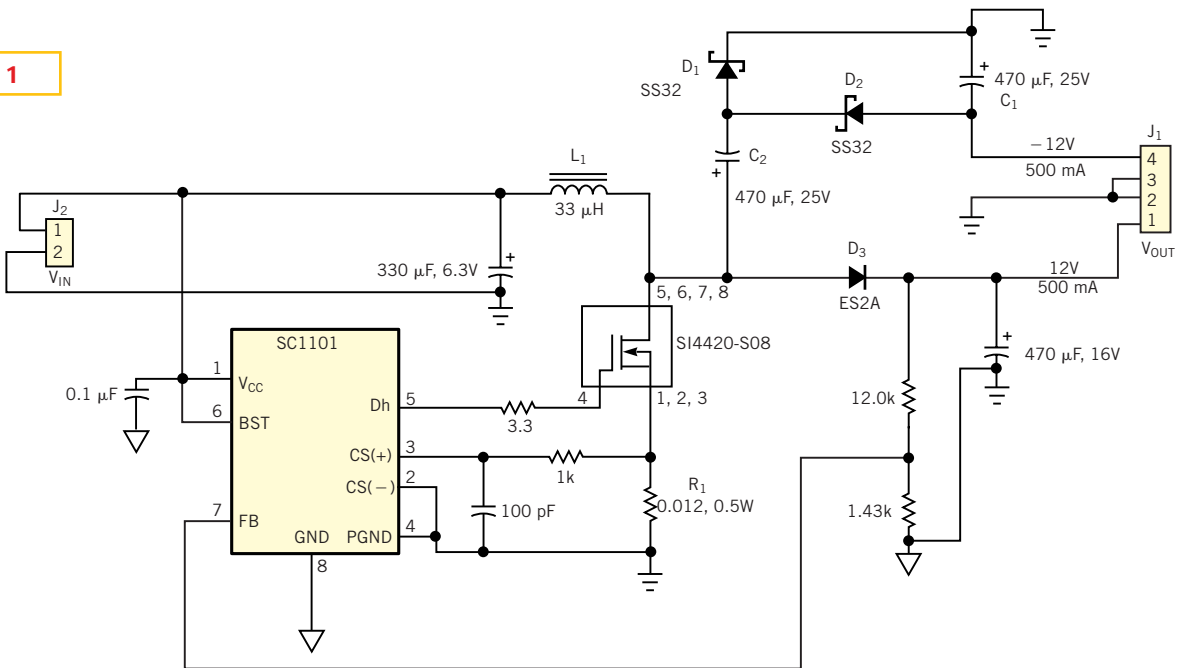
A fast silicon rectifier, D_3 , rather than a Schottky diode, rectifies the 12V output. Use of the silicon rectifier balances the voltage drops in the -12V rectifier circuit, which is two Schottky-diode drops, with the 12V rectifier drop. Because the application is power-limited, the circuit can trade off current that the -12V supply draws for current on the 12V output. If

the -12V output is unused, the 12V output can deliver 1A, and you can eliminate D_1 , D_2 , C_1 , and C_2 . To improve efficiency under this condition, you can also use a Schottky diode for D_3 . The controller provides separate grounds for power drive reference (PGND) and analog-circuitry common (GND). These two grounds must connect at the controller.

With dual outputs, the circuit's measured efficiency is 91% with $V_{IN}=5\text{V}$ and 93% with $V_{IN}=5.5\text{V}$. In **Figure 1**'s circuit, L_1 consists of a #T37-52 core (Micrometals Inc, www.micrometals.com) with 34 turns of 26-gauge wire. For applications that exceed ambient temperatures of 50°C , you can use a slightly larger core, such as a #T38-52, or a Kool MU core material available from Magnetics Inc (www.mag-inc.com). (DI #2279)

To Vote For This Design,
Circle No. 373

Figure 1



Tying a buck converter's BST pin to V_{CC} and the PGND pin to circuit ground configures the converter for boost operation.

Circuit eases three-phase monitoring

Henno Normet, Tavares, FL

Measuring line-to-line voltages in a delta-connected three-phase system can present special problems. Because all three lines may be floating several hundred volts above ground, you can not use nonisolated, grounded oscilloscopes or other single-ended instruments. Special isolation amplifiers are available for oscilloscopes, but they can cost several thousand dollars. You still need to make three measurements even with proper instrumentation. The circuit in **Figure 1** reduces the magnitude of the line-to-line voltages and combines them into one ground-referenced signal (**Figure 2**) that you can safely monitor with a grounded oscilloscope.

Phase C serves as a floating common (ground) for the circuit. Unity-gain buffer amplifiers IC₁ and IC₂ prevent loading of the 30-to-1 voltage dividers connected from A to C and from B to C. Op-

amp IC₃ is a differential-input instrumentation amplifier that provides a signal proportional to the voltage between A and B. Unity-gain amplifier IC₄ inverts the B-C signal such that the half-wave-rectified signals can combine with the proper 120° phasing. The forward voltage drops across D₁ to D₃ cause a small error. You can minimize this error by using germanium diodes (1N34s), which have lower voltage drops than their silicon counterparts, and by keeping the signal levels as high as possible.

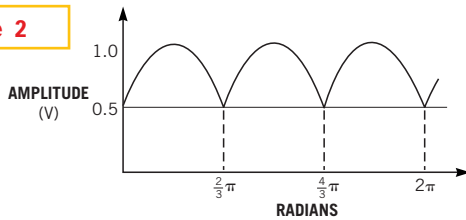
All the circuitry floats at power-line po-

tential; thus, it is dangerous to monitor the rectifier outputs. To obtain a safe output, you should use an ISO122P isolation amplifier. The ISO122P is a unity-gain amplifier with an output that is fully isolated from its input. You need two line-isolated ±15V supplies to power the ISO122P and the op amps. The circuit has additional applications, the details of which are beyond the scope of this Design Idea. For example, in some cases, it is important that the three phase voltages are equal (balanced); their absolute values may be of less importance. It is much easier to detect an imbalance

looking at a single waveform than it is looking at three waveforms. For continuous unattended monitoring, the design needs only one under-voltage/over-voltage detector, rather than one detector for each of the three phases. (DI #2283).

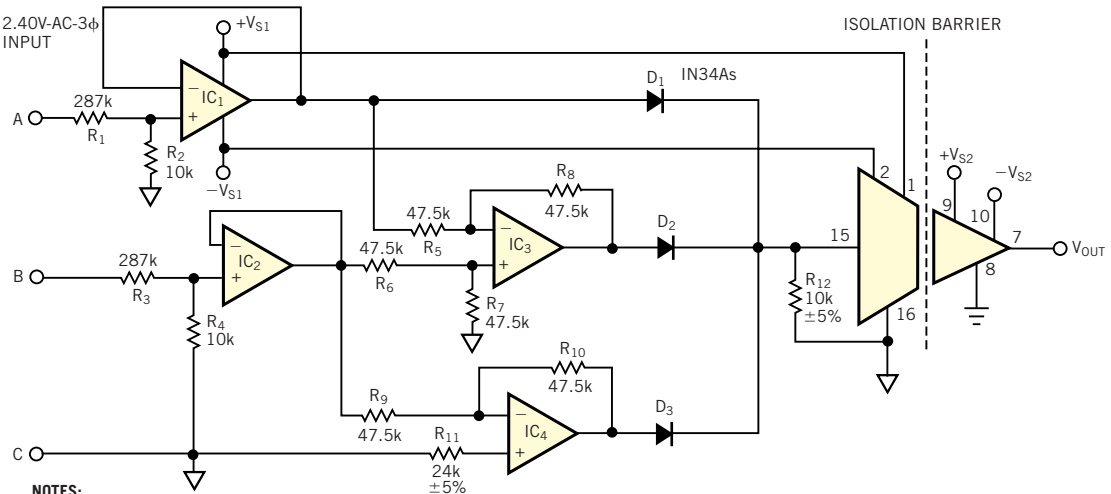
To Vote For This Design,
Circle No. 374

Figure 2



The output of the isolation amplifier in Figure 1 is a low-level, ground-referenced voltage.

Figure 1



- NOTES:**
- ▽ = FLOATING GROUND.
 - ⊥ = SYSTEM GROUND. (DO NOT TIE THESE POINTS TOGETHER.)
 - ALL RESISTORS EXCEPT R₁₁, R₁₂ = ±1% METAL FILM.
 - ALL OP AMPS = 1/4LM324.
 - ISOLATION AMPLIFIER = BURR-BROWN ISO122P.
 - CAUTION: LETHAL VOLTAGES ARE PRESENT IN THE CIRCUIT. USE EXTREME CARE.

A quad op amp combines the three half-wave-rectified phase voltages into one voltage for easy monitoring.

PLL forms simple MSK demodulator

Tom Napier, North Wales, PA

In minimum-shift-keying (MSK) signaling, two frequencies that differ by the bit rate represent a one bit and a zero bit. Normally, the frequency shift occurs at the peak of a cycle, so that neither the amplitude nor the slope of the waveform shows a discontinuity. We needed to transmit 300-baud ASCII text using ultrasonic transducers. These devices have a very narrow bandwidth around their 25-kHz resonant frequency, making MSK the obvious choice for modulation. A zero bit becomes 84 cycles of 25.2 kHz, and a one bit is 83 cycles of 24.9 kHz. It is easy to generate this signal with a PIC μ C and an 8-bit DAC. However, a traditional MSK demodulator circuit uses a center-frequency VCO and several mixers and filters. This design needs something simpler: to wit, the circuit in **Figure 1**.

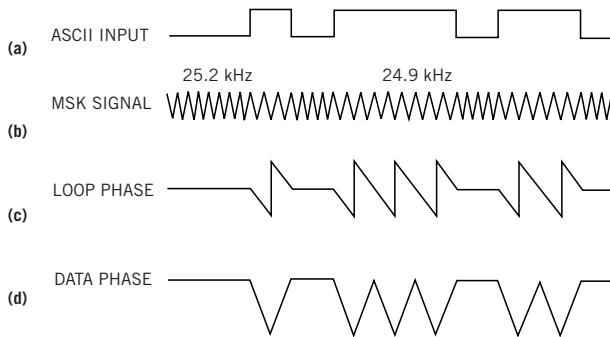
Because the transmitter sends 25.2 kHz between characters, the design phase-locks a 74HCT4046 PLL chip to this frequency. The chip has three phase detectors, each with different characteristics. By choosing

the correct two, you can demodulate the MSK input without losing phase lock on the zero-bit carrier. Phase Detector 3 on the PLL chip has a 360° linear range. That is, its mean output varies from 0 to 5V and then switches back to 0V as the phase passes through 360° . Adjust the frequency of the PLL chip so that it locks to 25.2 kHz

with a 180° phase error and an output of 2.5V. If the oscillator frequency remains fixed, then you can recognize a one bit by its phase error, which swings from 0 to 360° during the bit.

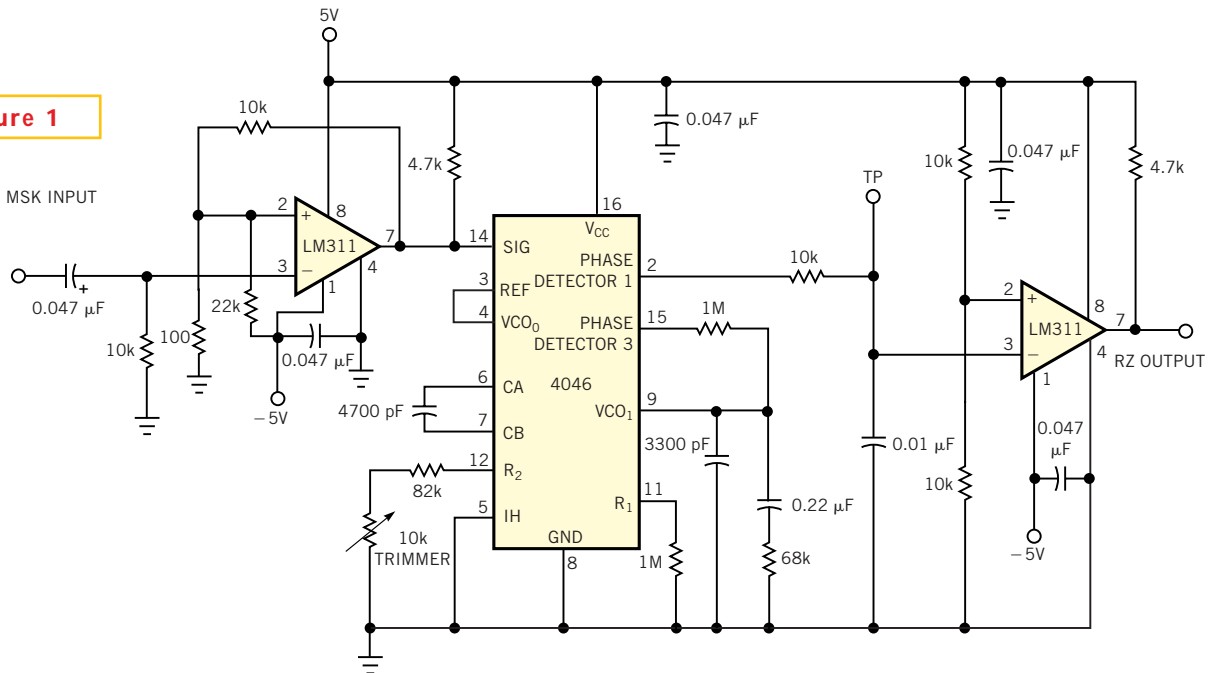
Because the initial lock is at the 180° point, a one bit results in a phase error that goes from 180° down to 0° . It then jumps

Figure 2



The ASCII input character for the letter "g" (a) produces a frequency-varying MSK signal (b). The phase error as seen by the loop (c) produces the phase-error signal at TP (d) that drives the output.

Figure 1



A PLL makes MSK demodulation inexpensive and easy.

to 360° and continues back down to 180°. The net result is a ramp that goes down to 0V, a jump to 5V, and then another down-going ramp. The mean voltage is 2.5V. Because the loop bandwidth is approximately 15 Hz, the instantaneous effect on the VCO is small, and the net frequency change is zero. Rather than detect the

ramp-jump-ramp waveform, use Phase Detector 1 as the data output. Because this block is a simple exclusive-OR gate, each one bit appears as a spike going from 5 to 0V and back. A comparator can change it into a return-to-zero version of the input signal. Alternatively, the output can go to a retriggerable monostable with a 3.5-msec

period to generate a good approximation of a nonreturn-to-zero output. **Figure 2** shows the circuit waveforms that occur for the letter “g.” (DI #2284).

To Vote For This Design,
Circle No. 375

μC makes inexpensive sine-wave generator

Jorge Luis B Romeu, IdeaWorks LTD, Syracuse, NY

You can use A/D converters or external, controllable oscillators to generate sine waves from low-power, low-cost μCs. However, these methods add cost, reduce reliability, increase circuit and software complexity, increase power consumption, and increase overall size. Alternatively, and with just a few lines of code, most μCs can easily generate multiple discrete sine waves. The example in **Figure 1** uses a 68HC705J1A to generate sine waves of 9 to 20 kHz. The circuit uses the μP’s square-wave output and switches between multiple RC filters of varying cutoff frequencies to achieve outputs with reasonable spectral purity.

The necessary code consists of a simple subroutine that can adapt to different needs, such as tone duration and multiple (sequential) tone output (**Listing 1**). Moreover, the generated frequency is based on a variable, “frec,” that a previous routine can pass to this subroutine. The duration time is based on a timer, which eliminates calculations of tone duration based on the cycle period. The code in **Listing 1** is for illustrative purposes, but you can use it as-is with proper headers. (You can download **Listing 1** and an example calling subroutine from *EDN*’s

Web site: www.ednmag.com. At the registered-user area, go into the Software Center to download the file from DI-SIG, #2278.)

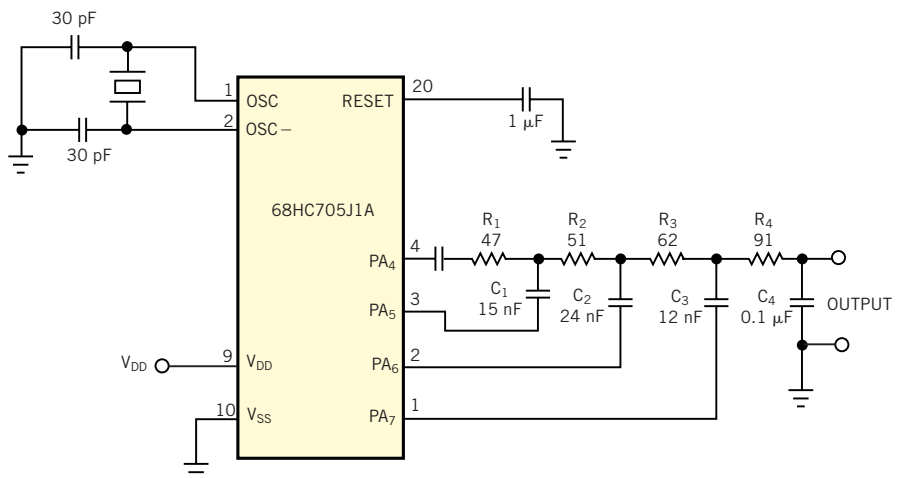
The basic circuit in **Figure 1** uses only

four output pins of the μC; three pins are for filter returns, and the fourth is the μP output. The filter constants are such that the highest cutoff frequency corresponds to the highest generated frequency. The

TABLE 1—EXAMPLE FREQUENCIES AND CORRESPONDING NUMBER OF LOOPS

Filter number and cutoff frequency (kHz)	Number for variable, "Frec" (=number of loops)	Actual output frequency (kHz)	Filter number active (R#, C#)
1: 10	14	9.5	1,2,3,4
2: 12	11	11.75	2,3,4
3: 15	8	15.35	3,4
4: 18	7	17.1	4 (always on)

Figure 1



Based on a simple subroutine, a μC can easily generate multiple sine waves from 9 to 20 kHz by switching external RC filters on and off.

subsequent cascaded filters cut off at lower frequencies, and the circuit can switch these filters in or out depending on the desired output (Table 1). The μC 's speed, code efficiency, the number of discrete frequencies to be generated, and the spacing between those frequencies all determine the maximum frequency of operation. The number of loops for one-half cycle of output frequency (50% duty cycle) equals the number of cycles per loop \times time/cycle. With a 3.58-MHz crystal, the time per software cycle is 558 nsec. (The 705J1A μC uses one-half of the oscillation frequency for its internal operating frequency.) Table 1 shows some example frequencies and the corresponding number of loops.

The RC filters have a cutoff frequency, or -3-dB point, at the generated frequency for maximum linearity and minimum distortion. You should note that the out-

LISTING 1—FREQUENCY-GENERATING SUBROUTINE

```

fgen  sta temp1          ;store accumulator in a temp variable
      lda #04           ;number of interrupts for 250ms (5*65ms)
timer bset 2,tscr        ;clear real time interrupt flag
innr3 bset 4,output     ;turn on oscillator output
      ldx freq          ;number for one half cycle
innr1 decx              ;count down for half cycle
      bne innr1         ;if not finished, keep counting
      bclr 4,output     ;turn off oscillator output
      ldx freq          ;number for other half of cycle
innr2 decx              ;count down for half cycle
      bne innr2         ;if not finished, keep counting
      brclr 6,tscr,innr3 ;if 65ms not passed (see timer/counter)
section of docs) do another cycle
      decx              ;subtract from the five 65ms for 250ms
total bne timer        ;have 250ms gone by? If not go back and
repeat all
      lda temp1         ;restore accumulator
      rts
  
```

put is approximately the same level across all frequencies because the resistors are always in series when driving a high-impedance load. However, because of leakage when a capacitor is active, or grounded, the lowest frequency has a lower output than the highest frequency. You should optimize the R and C values in Figure 1 for your application, desired out-

put frequencies, and impedance. You can switch the filters on and off sequentially, independently, or in combinations to suit different needs. (DI #2278)

To Vote For This Design,
Circle No. 376

74ACT74 makes low-skew clock divider

Tom Napier, Consultant, North Wales, PA

Serial-data systems often generate an internal clock at twice the data rate for mid-bit sampling or for generating bi-phase codes. External equipment and some internal processes require a clock that runs at the data rate. Simply dividing the twice-rate clock with a flip-flop generates a data-rate clock that is skewed by one logic delay with respect to the input. This delay can be a significant fraction of the bit period. You can use specialized PLL-based low-skew divider chips to deal with this problem, but these chips have a limited frequency range and are not designed to follow rapid changes in the data rate.

The circuit in Figure 1 uses a dual flip-flop, the 74ACT74, to generate both clock rates as well as both clock polarities with negligible skew. One half of the chip, IC_{1A}, acts as a normal divide-by-two circuit. The other half, IC_{1B}, tracks the input clock because the input's leading edge triggers IC_{1B} high and the input's trailing edge resets IC_{1B}. The divider transitions are synchronous within a few hundred picoseconds with the positive transitions of the twice-

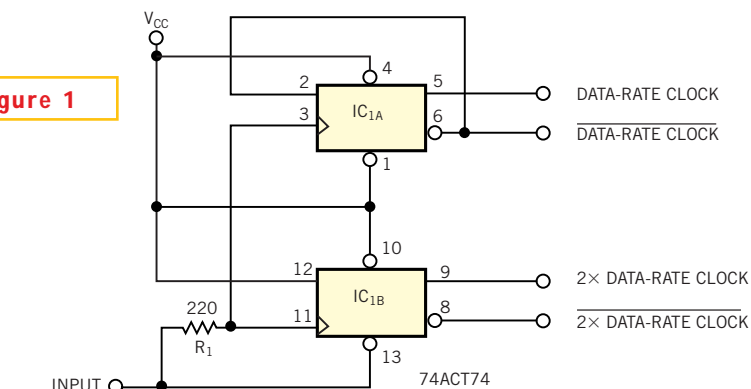


Figure 1

The 74ACT74 dual flip-flop generates two clock rates with negligible skew.

rate clock that the chip's other half generates. This circuit works with inputs from a few hertz to more than 100 MHz. Without R₁, the input removes the reset from the twice-rate flip-flop at the same moment as the input clocks this flip-flop on. Theoretically, this setup is allowable because the 74ACT74's reset-recovery time is

specified as 0 nsec. In practice, a resistor in the 100 to 500Ω region, in conjunction with the chip's input capacitance, delays the clock inputs slightly and adds a useful safety margin. (DI #2282)

To Vote For This Design,
Circle No. 377

Design Idea Entry Blank

Entry blank must accompany all entries. \$100 Cash Award for all published Design Ideas. An additional \$100 Cash Award for the winning design of each issue, determined by vote of readers. Additional \$1500 Cash Award for annual Grand Prize Design, selected among biweekly winners by vote of editors.

**To: Design Ideas Editor, EDN Magazine
275 Washington St, Newton, MA 02158**

I hereby submit my Design Ideas entry.

Name _____

Title _____

Phone _____ **Fax** _____

E-mail _____

My e-mail address may be published Yes No

Company _____

Address _____

Country _____ **ZIP** _____

Design Idea Title _____

Social Security Number _____

(US authors only)

Entry blank must accompany all entries. (A separate entry blank for each author must accompany every entry.) Design entered must be submitted exclusively to EDN, must not be patented, and must have no patent pending. Design must be original with author(s), must not have been previously published (limited-distribution house organs excepted), and must have been constructed and tested. Fully annotate all circuit diagrams. Please submit software listings and all other computer-readable documentation on a IBM PC disk in plain ASCII.

Exclusive publishing rights remain with Cahners Publishing Co unless entry is returned to author, or editor gives written permission for publication elsewhere.

In submitting my entry, I agree to abide by the rules of the Design Ideas Program.

Signed _____

Date _____