Loading PLL Frequency Synthesizers with COP8SAx7

National Semiconductor
Application Note 1098
Steven Goldman
June 1998

1.0 INTRODUCTION

This application note describes the hardware and software required for loading Single or Dual Frequency Synthesizers for RF Communications with National Semiconductor's COP8™ family of microcontrollers. COP8SAx7 is an excellent choice for a low cost, single chip solution for configuring all parameters necessary for the frequency synthesizer.

2.0 DESIGN REQUIREMENTS

This design example uses the LMX2320 Frequency Synthesizer and the COP8SAA7 Microcontroller. National's family of PLLatinum™ Frequency Synthesizers requires several registers to be loaded via a serial interface. (Please refer to the datasheets for the appropriate LMX PLL used in your design.)

Registers to be configured include the Programmable Reference Divider (R Counter), the Prescaler Select (S Latch), and the Programmable Divider (N Counter). Data being sent to these registers are organized in "data streams." The datasheets for various PLL's will define the details, and bit placement, for these data streams.

The microcontroller must be able to transmit several data streams, control the LE (Load Enable) signal, and generate a clocking source. These "data streams" will be padded with leading zeroes, such the data streams are an integer multiple of eight (8, 16, 24, etc.) bits.

After programming the PLL, the microcontroller will be placed in a power down mode, drawing virtually zero current from the circuit. The software is easily adaptable to allow the user to include many other functions.

3.0 FEATURES OF THE COP8SAx7

The COP8SAA716M9 was selected as the lowest cost solution. This is the 16-pin SOIC member of the COP8SAx7 One Time Programmable (OTP) family. Other COP8 family members would utilize the same source code.

Several features of the COP8SAx7 are utilized in this design:

- Power up Reset
- Internal R/C
- Internal Pull-Up Resistors
- Power Down Mode
- MICROWIRE™ Port
- High Current Outputs

3.1 Power Up Reset

COP8SAx7 includes internal reset circuitry. This eliminates the need for the traditional diode, capacitor, resistor circuit common to other COP8 family members, and other microcontrollers. Internal reset circuitry saves the system designer valuable board space and the expense of extra components. When the OTP COP8SAx7 is programmed, the "Internal Reset" option must be selected.

The Application Schematic, *Figure 1*, shows the connection from the $\overline{\text{RESET}}$ pin to V_{CC}. When programmed to utilize the internal circuit, the $\overline{\text{RESET}}$ pin must remain HIGH. Refer to the COP8SAx7 Datasheet for further details.

3.2 Internal R/C

COP8SAx7 includes an internal R/C oscillator circuit, eliminating the need for an external clock source, Crystal resonator, or external resistor and capacitor. This will save the system designer additional board space, and the expense of additional components. Since the serial interface is synchronized with a clock signal (CLOCK), the internal clock of the COP8SAx7 does not need to be excessively stable over temperature or time. Changes in clock stability are therefore ignored. The circuit can easily be modified to accept either clock input, external crystal or external R/C up to 10 MHz. Please refer to the COP8SAx7 Datasheet for further details.

When programming the OTP COP8SAx7, please ensure that the "Internal R/C" option is selected.

3.3 Internal Pull-Up Resistors

COP8SAx7 includes weak pull-up resistors on all input/output ports. These are approximately 100k in value. The application circuit includes a jumper block (JP1) which selects the "DIAGNOSTIC" mode.

The appropriate bit in the CONFIGURATION register for PORTG must be set to "0", to define that pin for input. The internal pull-up option is configured by writing a "1" to the corresponding bit in the DATA register for PORTG. For the application schematic shown this would be G0.

Utilizing the internal pull-up once again saves the system designer the expense of an additional resistor on the final PCB, simplifies layout, and saves valuable board space. When the pin is left "floating", the software will read this input as "1." When JP1 is moved to GND, the pin will read as "0."

3.4 Power Down Mode

After the software runs completely through, the COP8SAx7 will be placed in power down mode. Power consumption varies depending on the COP8 selected, but will be in the microamps range. Virtually zero.

The system designer may choose to eliminate the power down aspect of the software provided. For many applications, COP8SAx7 will be performing other system tasks.

3.5 MICROWIRE Port

Utilizing the MICROWIRE port, on the COP8SAx7, is the real heart of this design. All COP8 devices include on-board MICROWIRE hardware. This port consists of SI (Serial In), SO (Serial Out), and SK (Serial Clock). The SI pin is *not* used for this design. Details on operating MICROWIRE can be found in the COP8 Databook. A brief description is found below.

Some LMX PLL's require a "data stream" of 22 bits, which have been rounded up to 24 bits, or 3 bytes. Each byte is loaded into the Serial Input/Output Register (SIOR). Setting the BUSY flag of the COP8's control register will initiate MICROWIRE transfer. Contents of the SIOR are then shifted out of pin G4/SO one bit at a time. The MICROWIRE hardware will also generate 8 clock pulses on the CLK pin (G5/SK). After 8 bits have been shifted, the BUSY flag will be reset.

The application software will configure the CONTROL register to ensure that MICROWIRE is set to the proper mode. Subroutine "µWire" takes care of all functions needed to transfer data from the Accumulator, to the SIOR Register, and finally out of the MICROWIRE Port. A separate pin of PORTG (G3) controls the LE signal required by the PLL. This is because the LE line must be kept low for several data bytes. The sequence of sending the correct number of bytes completes one data stream. Depending on the PLL used several data streams will be sent.

3.6 High Current Outputs

COP8SAx7 outputs supply sufficient current to directly drive LED's. This application circuit utilizes one LED to provide visual feedback after the configuration sequence has successfully completed.

4.0 HARDWARE CONSIDERATIONS

The application schematic shows the connections from the LMX2320 to the COP8SAA7. Other LMX PLL's will have similar connections to the LE, DATA, and CLOCK signals. The Power Down (PD) pin of the LMX2320 was not utilized in this design, but could easily be connected to an unused I/O pin of the COP8SAA7. Output pins will remain in their last known state before Power Down Mode was entered.

Please refer to the Datasheet for your choice of PLL, to determine the VCO, Loop Filter, etc., that surround the frequency synthesizer portion of the design. This schematic is intended only to show the connections to the microcontroller.

4.1 Load Enable (LE)

G3 is used to generate the LE signal for the PLL. This signal is pulled low to initiate the start of programming the PLL. For this example, after 3 bytes have been sent the software will bring LE high. As stated earlier, the number of bytes sent, while LE remains low, will vary.

4.2 Diagnostic Mode

This mode has been added to aid the system designer during the early stages of design. After programming the PLL, pin G0 (DIAGMODE) will be sampled and checked for HIGH (left unconnected, internally pulled-up) or LOW (tied to GROUND). When this input is LOW, the device will enter the DIAGNOSTIC mode.

While in "Diagnostic Mode" the data streams will be continuously sent to the PLL and the timing diagram will be as shown (repeating). Note the use of TRIGGER to signify the start and end of configuration (See *Figure 2*).

4.3 Scope Trigger Output/LED

The signal TRIGGER (G2) has two purposes. System designers may choose to eliminate this feature in their final production level product. During the design phase it will have great value.

The first purpose will be a connection to an LED. After the PLL has been loaded, TRIGGER will be pulled low for less than one second, turning on an LED (shown on the schematic as D1). This is long enough for the LED to be seen by the naked eye. If you do not see the LED lit, you will know that the software did not run (or your prototype is wired incorrectly).

The second purpose for the TRIGGER signal is to indicate the start and end of the configuration sequence. TRIGGER will drop LOW at the start of the sequence, and is raised HIGH after all data streams have been sent. The timing diagram clearly shows this process. (See *Figure 2*.)

While in the DIAGNOSTIC mode, the data streams will be continuously sent to the PLL. The falling edge of TRIGGER can be used to trigger an oscilloscope. LE (Load Enable signal) would not serve as an appropriate trigger source to an oscilloscope because multiple "different" data streams are sent. A storage scope would be needed to capture data triggered on the edge of LE. While utilizing the TRIGGER signal, no storage scope is required. This is very helpful, since quite a few of us do not have storage scopes.

4.4 Reset Circuitry

COP8SAx7 includes an internal Power Up Reset circuit. To utilize this feature connect RESET to V_{CC} as indicated in the schematic. (Other COP8 devices may not include an internal reset circuit.) No other reset hardware is necessary. When you program the COP8SAx7 OTP, make certain that the Internal Reset option is selected.

4.5 Prototype

Evaluation boards for LMX PLL's are available from National. These boards offer connections to CLOCK, DATA, LE, power, and reference signals. Specify the correct LMX part when ordering an evaluation board.

LMX evaluation boards can be programmed from the parallel port of a Personal Computer or from a COP8 based prototype board.

The COP8 prototype can easily be built on any type of breadboard, following the application circuit shown. Alternately, the COP8-EVAL-HI01 (COP8 Evaluation Board) is available from National. This board is built and tested, with additional peripherals such as Temp Sensor, EEPROM, and A-to-D on board.

With either COP8 target board, the appropriate signals (CLOCK, DATA, and LE) must be connected to the LMX evaluation board. There are special lines of assembly language code (clearly marked) that enable the software to run on the COP8-EVAL-HI01. An LED on the COP8-EVAL-HI01 is used in place of "D1", and a pushbutton (SW3) replaces "JP1."

5.0 SOFTWARE CONSIDERATIONS

The software included was written with the primary goal of allowing the RF designer the ability to configure the PLL. Please note, this is the RF designer, not a team of software engineers. The RF designer will also be able to maintain the software. Many companies have separate Hardware and Software departments, particularly those involved in wireless design. This is because RF design has always been a special branch of Electrical Engineering. The COP8SAx7 adds very little additional cost to the system, requires zero external components, and gives design control to the RF designer. System designers may find that the COP8 could replace other circuitry in their design.

The only portion of the software that needs to be customized is the "DATA" section. It is in this section that the system designer inserts the parameters that the LMX PLL's require.

Although the software is amply commented, additional remarks are found below. For further details, refer to the flowchart, *Figure 3*.

The system designer will need some type of OTP programmer in order to program the COP8SAx7 devices. These include:

- COP8SA-EPU
- COP8SA-DM
- Conventional EPROM programmers from Data I/O, BP Microsystems, etc.

Please check your equipment for compatibility with the COP8SAx7 family.

5.1 Declaration Section

The "DECLARATION" portion of the assembly file contains the addresses of the registers (Memory Map) within the COP8. If the system designer uses a device different than the COP8SAx7, these addresses may change. Alternately, an "include" file may be used. However, when the Declaration Section is placed at the top of the assembly source file, additional documentation is not necessary.

5.2 Loading the Memory of LMX PLL

The main section of the software starts at the label "LOADPLL." At the start of the programming sequence, pin 2 of PORTG is given an RBIT (Reset Bit) command. This will drive the signal called "TRIGGER" low. Later, we will bring TRIGGER high to signify the conclusion of the programming sequence. This was discussed further in the Diagnostics Mode Section.

The program then goes to the section of code labeled "Data Storage Area" to upload various parameters needed to configure the PLL. These include:

- Number of Data Streams
- Bytes per Data Stream
- Data Streams

The number of data streams to be sent (called "STREAMS") is loaded into register R1. Register R1 will keep track of how many data streams were sent.

The number of bytes per data stream (called "BYTESPER") is loaded into register R3. Register R3 will keep track of how many bytes per data stream have been sent out.

The actual "data streams" are stored in COP8 memory starting at the label "DATA." (Refer to the examples in the source code.)

Next, the Load Enable (LE) signal is sent LOW. The falling edge of LE initiates the internal shift registers of the PLL. The LAID (Load Accumulator Indirect) instruction is used in conjunction with the B pointer to step through the lookup table. LE will be driven HIGH when each data stream completely loads.

At this point the software determines if more data streams need to be sent. If so, we return to the DATA section, and upload the next Data Stream.

When all data streams have been sent, the TRIGGER output is driven HIGH.

5.3 Entering the DIAGNOSTIC Mode and Power Down

After completely configuring the PLL, we determine if we should enter the DIAGNOSTIC mode. We then read PORTG pin 0 (the DIAGMODE signal), to see if it was left floating (HIGH), or pulled to ground (LOW). The IFBIT test will be TRUE if this pin was left floating, and the next command "JMP HALTMODE" will execute. When IFBIT is false, the next statement is skipped.

While in the DIAGNOSTIC mode we return to LOADMEM and perform all steps again. Since TRIGGER will go high only once during the sequence, it is ideal to use as a trigger source for an oscilloscope. During the DIAGNOSTIC mode, you may see additional noise on the output of the PLL, due to the constant reloading of parameters.

When DIAGNOSTIC mode is exited, or not used at all, we enter the Power Down Mode, or HALT mode. Before powering the COP8SAx7 completely down, TRIGGER is driven low for about one second. Normally, this output would be connected to an LED. The intent is to provide one pin that can very easily be monitored without elaborate test equipment. Writing a "1" to PORTGD, bit 7, will place the device in HALT mode. The compiler will issue a warning about this line. Do not be concerned about this warning. Otherwise, the program should assemble with zero errors.

If you prefer not to use the HALT mode, make sure that you include some sort of "HERE: JMP HERE" statement so that the microcontroller does not wander off into never-never land.

5.4 Customizing the Software

LOADPLL.ASM assembly file has been included. This file is also available for download from *www.national.com*.

The section labeled "Data Storage Area" needs to be modified by the system designer. System parameters are calculated by following formulas given in the LMX datasheet, or by utilizing the PLL software available from National. (Make note of the relative positions of the MSB and LSB for the Data Streams.) Either way, these parameters need to be inserted into the source code.

These parameters are:

- STREAMS
- BYTESPER
- DATA

STREAMS: The number of data streams to be sent. This value will vary depending on the PLL chosen, but will normally be "2" for a single frequency synthesizer, "4" for a dual. Fractional-n parts and other PLL's will vary. Simple modification of that value is all that is needed. Register R1 will keep track of how many data streams were sent.

BYTESPER: The number of bytes per data stream. For this example there will be three bytes per data stream. Future LMX devices may require more/less bytes per data stream. Changing this parameter will accommodate various length data streams. Register R3 will keep track of how many bytes per data stream have been sent out.

DATA: The actual data stream(s). Most LMX PLL's require 22 bits of data. This data stream is rounded up to the nearest multiple of eight. In this case, 24 bits, or three bytes would be sent. As discussed earlier, sending out packets of eight bits each is more natural for the MICROWIRE interface. Remember to pad the most significant bits (MSB) with leading zeroes. Extra bits are ignored by the LMX PLL's. The internal PLL shift registers only the last bits sent.

LE will be driven HIGH when each data stream completely loads.

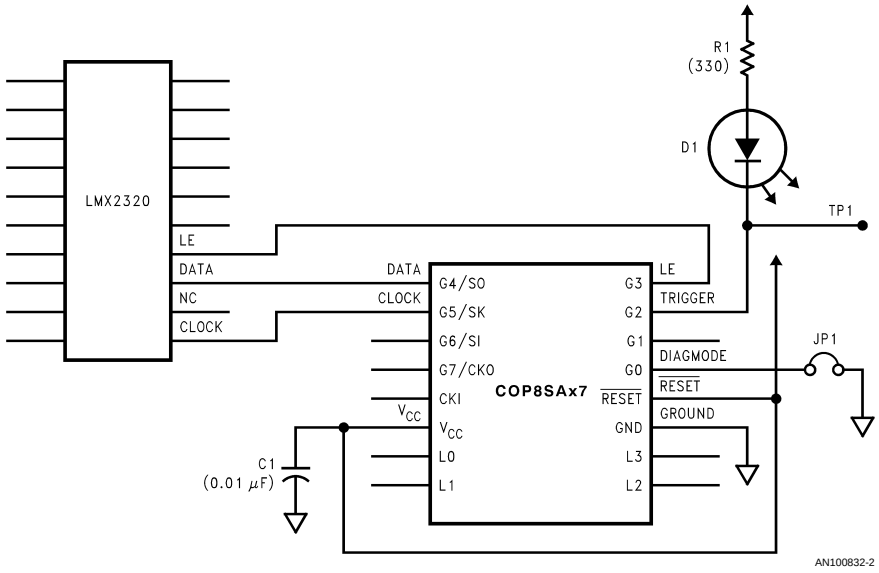
6.0 CONCLUSIONS

National's family of LMX PLL's is an excellent choice for RF designers. The COP8SAx7 is a low cost, easy-to-use microcontroller for loading configuration data. Utilizing the infor-

mation above, the RF designer can successfully incorporate a microcontroller into the frequency synthesizer design, without previous microcontroller design experience.

7.0 REFERENCES

1. National Semiconductor, “National Products for Wireless Communications-1996”, LMX2320/LMX2325 PLLatinum Frequency Synthesizer for RF Personal Communications Datasheet, pages 1–42 to 1–59.
2. National Semiconductor, “National Products for Wireless Communications-1996”, Introduction to Single Chip Microwave PLLs, Application Note 885, pages 1–149 to 1–153.
3. National Semiconductor, “COP8 Microcontroller Databook” 1996/1997.
4. COP8 Feature Family User’s Manual
5. 1997 Wireless Databook



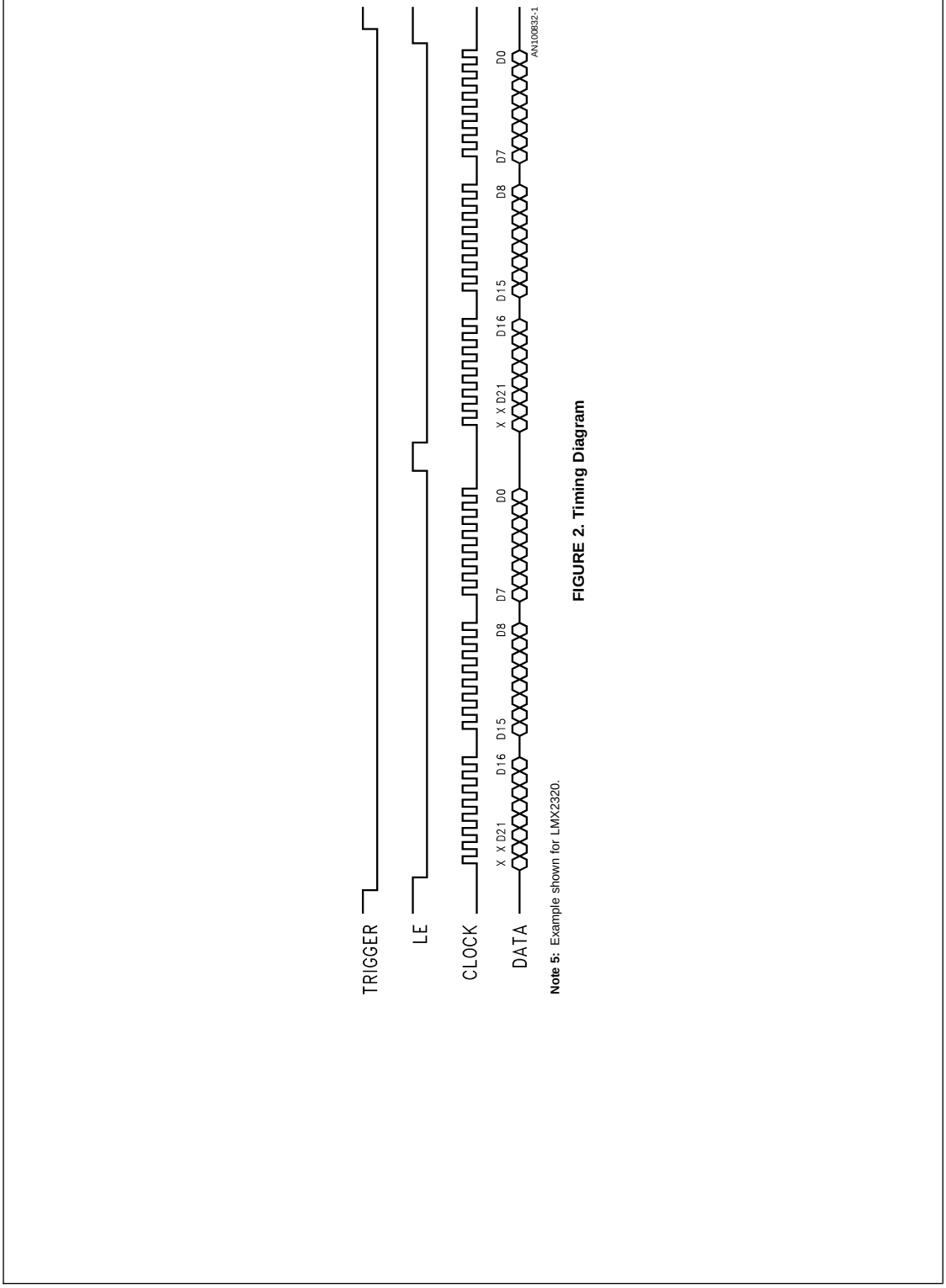
Note 1: Install Jumper #1 (JP1) to initiate the DIAGNOSTIC mode.

Note 2: Remove JP1 for normal operation.

Note 3: Test point #1 (TP1) is used to trigger an oscilloscope. Refer to text for details.

Note 4: LED optional. Remove R1 and D1 if not needed.

FIGURE 1. Application Circuit Example - LMX2320 and COP8SAx7



Note 5: Example shown for LMX2320.

FIGURE 2. Timing Diagram

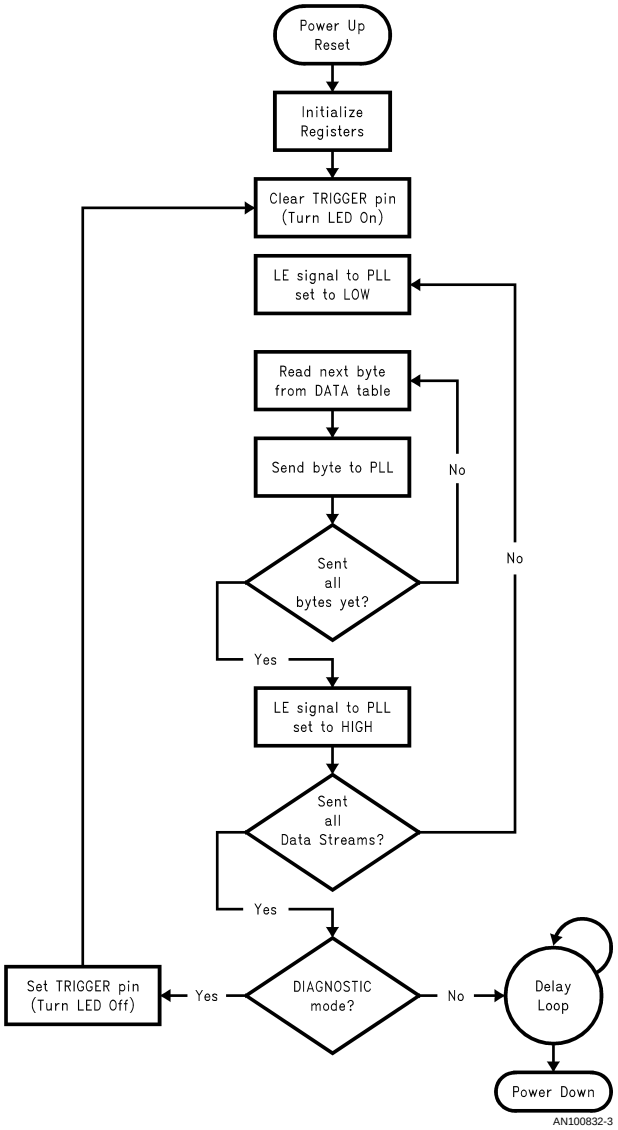
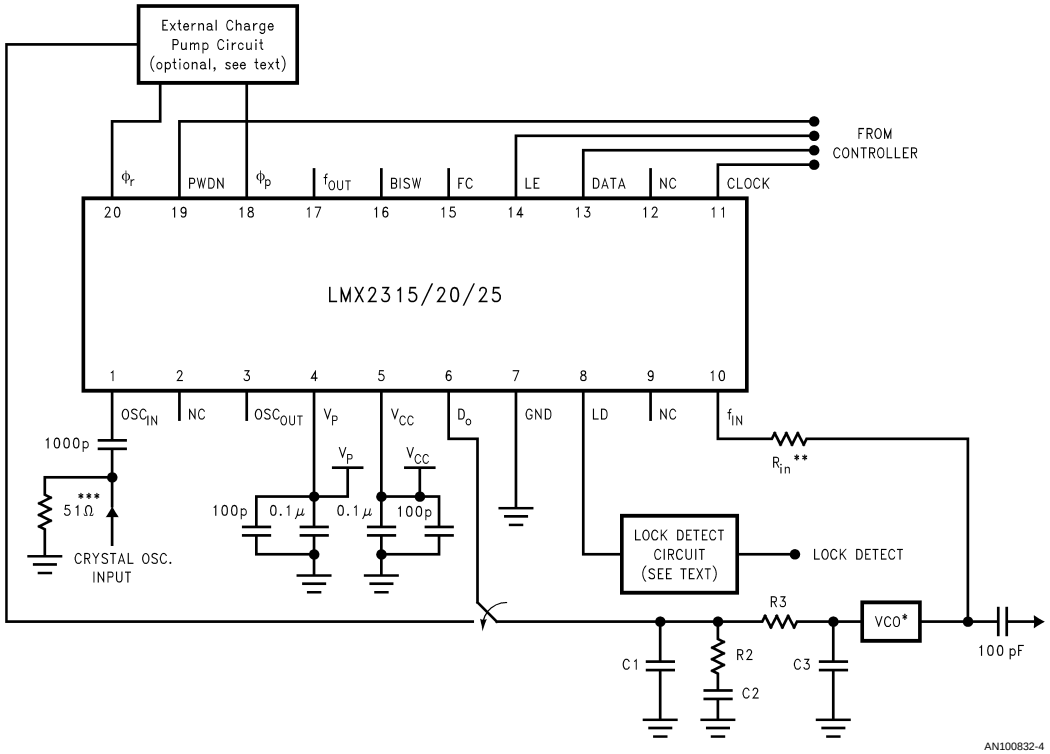


FIGURE 3. LOADPLL Program Flow Chart (Single Frequency PLL)

SCHEMATIC


External Charge Pump Circuit (optional, see text)
FROM CONTROLLER
ϕ_r
PWDN
ϕ_P
f_{OUT}
BISW
FC
LE
DATA
NC
CLOCK
20 19 18 17 16 15 14 13 12 11
LMX2315/20/25
1 2 3 4 5 6 7 8 9 10
OSC_{IN}
NC
OSC_{OUT}
V_P
V_{CC}
D_o
GND
LD
NC
f_{IN}
1000p
***
51Ω
CRYSTAL OSC. INPUT
V_P
V_{CC}
100p
0.1μ
0.1μ
100p
R_{in} **
LOCK DETECT CIRCUIT (SEE TEXT)
LOCK DETECT
R3
VCO*
100 pF
C1
R2
C2
C3
AN100832-4

```
;=============================================================================
;
;                                LOADPLL.ASM
;
;                This software accompanies an application note
;             demonstrating the loading of configuration data to an
;                           Single Frequency Synthesizer (PLL)
;                                such as the LMX2320.
;
;                               Rev 1.6, January 1998
;               (Configured for COP8SAA, may be adapted to any COP8)
;
;                               by:  Steven Goldman
;                                    National Semiconductor
;                                    Senior Field Applications Engineer
;
;
;
;
      .TITLE LOADPLL
      .CHIP  8SAA
      .SECT  MAIN,ROM,ABS=0
;
;
;
; Declarations (Assign Symbolic names to Addresses)

         PORTGD = 0xD4         ; PORTG Data Reg
         PORTGC = 0xD5         ; PORTG Config Reg
          PORTG = 0xD6         ; PORTG Register (Input Only)

          PORTD = 0xDC         ; PORTD - is only available on
                               ; larger COP8 devices.
                               ;

          CNTRL = 0xEE         ; Control Register
            PSW = 0xEF         ; Processor Status Word
             R0 = 0xF0         ; Loop Counter 0
             R1 = 0xF1         ; Loop Counter 1
             R2 = 0xF2         ; Loop Counter 2

           SIOR = 0xE9         ; MICROWIRE Register
         STKPTR = 0xFD         ; Stack Pointer

;-----------------------------------------------------------------------------
;
; Setup Control Registers, I/O Ports, Interrupts, etc.
;
START:                         ;
                               ;
         LD STKPTR, #0x2F      ; Initialize Stack Pointer
                               ;   (Address #0x2F for COP8SAA)
                               ;   (Address #0x6F for COP8SAC)

         LD CNTRL, #0x09       ; CNTRL is the control register
                               ;   for Timer1 and MICROWIRE/PLUS
                               ;   #0x09 = (0000:1001)
                               ;
                               ;   "T1C3" (Bit7) = 0
                               ;   "T1C2" (Bit6) = 0
                               ;   "T1C1" (Bit5) = 0
                               ;   "T1C0" (Bit4) = 0
                               ;   "MSEL" (Bit3) = 1, Enables SO and SK
                               ;                      for their MICROWIRE
                               ;                      functions
                               ;   "IEDG" (Bit2) = 0, Ext.Int.Edge Select
                               ;   "SL1"  (Bit1) = 0
```

```
                                ;   ″SL0″  (Bit0) = 1, MICROWIRE divider is
                                ;                      selected by both
                                ;                       SL1 and SL0. In this
                                ;                      example Divide-by-4,
                                ;                       (SL1:SL0 = 0:1),
                                ;                       is used.
;
;
;
; Initialize PORTG's CONFIGURATION Register for Input or Output
;
;   NOTE: I/O's that are not connected on the circuit diagram
;         are marked ″NC″ in the following tables.
;

          LD PORTGC, #0x3C      ; #0x3E = (0011:1100)
                                ; 1=Output, 0=Input
                                ;
                                ;
                                ;   Signal Name  Pin  Function
                                ;   -----------  ---  --------
                                ;   ″NC″         (G7) Input
                                ;   ″NC″         (G6) Input
                                ;   ″CLOCK″      (G5) Output
                                ;   ″DATA″       (G4) Output
                                ;   ″LE″         (G3) Output
                                ;   ″TRIGGER″    (G2) Output
                                ;   ″NC″         (G1) Input
                                ;   ″DIAGMODE″   (G0) Input
;
;
;
; Initialize PORTG's DATA Register for LOW, HIGH, Hi-Z, or Pull-Up

          LD PORTGD, #0x05      ; #0x05 = (0000:0101)
                                ; 1=Output HIGH, or Input Pulled HIGH
                                ;
                                ;   Signal Name  Pin  Function
                                ;   -----------  ---- ------------
                                ;   ″CKO″        (G7) Input Hi-Z
                                ;   ″SI″         (G6) Input Hi-Z
                                ;   ″CLOCK″      (G5) Output LOW
                                ;   ″DATA″       (G4) Output LOW
                                ;   ″LE″         (G3) Output LOW
                                ;   ″TRIGGER″    (G2) Output HIGH
                                ;   ″NC″         (G1) Input Hi-Z
                                ;   ″DIAGMODE″   (G0) Input Pull-Up
;
;
;
;  End of Initialization Section
;
;-----------------------------------------------------------------------
;
;  LOADPLL.ASM (Main Program Section)
;
;

LOADPLL:  RBIT 2, PORTGD        ; Oscilloscope trigger, TRIGGER=LOW
          LD A, #STREAMS        ; Fetch number of ″Data Streams″
          LAID                  ;   from lookup table
          X A, R1               ; Use R1 to count number of
                                ;   ″Data Streams″ to send.
                                ;
          LD B, #0x00           ; Start with offset pointer=0
                                ;
CONFIG:   LD A, #BYTESPER       ; Get Bytes per Data Stream
```

```
          LAID                  ;   from lookup table
          X A, R0               ; Use R0 to count number of
                                ;   "Bytes per Data Stream"
                                ;
          RBIT 3, PORTGD        ; Guarantee that signal LE=LOW
                                ;
NEXTDATA: LD A, B               ; Load offset into "A"
          ADD A, #DATA          ; Add starting address of lookup table
          LAID                  ; Fetch data from lookup table
          JSRL uWire            ; Transmit via MICROWIRE
          LD A, [B+]            ; Increment B Register
          DRSZ R0               ; Have we finished this Data Stream yet?
          JMP NEXTDATA          ; Not yet, get another byte!
                                ;
          SBIT 3, PORTGD        ; After sending a complete "Data
                                ;   Stream", set LE=HIGH.
          DRSZ R1               ; Have we sent all the Data Streams?
          JMP CONFIG            ; No, send the next one!
                                ;
          SBIT 2, PORTGD        ; Set TRIGGER = HIGH, to signify
                                ;   that all Data Streams have been sent.
                                ;
          IFBIT 0, PORTG        ; Should we go to "Diagnostic Mode."
                                ;   This mode is started when the
                                ;   DIAGMODE pin (G.0) is pulled
                                ;   to GROUND.  When left floating,
                                ;   the internal pull-up resistor
                                ;   will drive DIAGMODE=HIGH,
                                ;   and the IFBIT command will execute
                                ;   the next instruction.
                                ;
          JMP HALTMODE          ; Powerdown the COP8, when DIAGMODE=HIGH
                                ;
          JMP LOADPLL           ; DIAGMODE = LOW, please repeat the
                                ;   loading process.
;
;
;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
;
; HALTMODE (LED turned on briefly before powerdown)
;
;
;  At this point the PLL will be fully configured.  TRIGGER (G.2) will
;  go low for 3/4 of one second.  This can easily be seen on
;  an oscilloscope, or connected to an LED, for visual indication
;  that the software has completed execution.
;
;  During prototyping, this a handy way to "see" things
;  working, without fancy test equipment.
;
;

HALTMODE: RBIT 2, PORTGD        ; Turn LED On (TRIGGER=LOW)
          JSR DELAY             ; Software time delay
          SBIT 2, PORTGD        ; Turn LED Off (TRIGGER=HIGH)
                                ;
                                ;
;--------------------------------
;         SBIT 0, PORTD         ; WARNING!  Remove the comment
                                ;   delimiter ";" only if COP8-EVAL-HI01,
                                ;   the COP8 Evaluation Board, is your
                                ;   target hardware.  The application
                                ;   circuit indicates an LED connected
                                ;   to G2.
;--------------------------------
                                ;
```

```
                              ;
          SBIT 7, PORTGD      ; NOTE: It is normal for ASMCOP (the COP8
                              ;   Assembler) to issue a warning, that
                              ;   Bit7 of PORTGD is being set.
                              ;
                              ;   Ignore that warning.  Therefore, the
                              ;   file should assemble with
                              ;   ERRORS: 0,   WARNINGS: 1
                              ;
                              ; COP8 will remain inactive,
                              ;   until RESET.
;
;
;
;-----------------------------------------------------------------------
;
; DATA STORAGE AREA
;
;   LOADPLL.ASM was designed and organized so that the following
;     is the only section of code requiring customization.
;     See the application note text for further explanation. The
;     software has been generalized wherever possible.  If LOADPLL.ASM
;     is used in a larger program, make certain that the LAID instruction,
;     and the DATA table, are located on the same page.  (LAID uses
;     only the lowest 8 bits of address, to locate the table.)
;
;     "Data Stream" is defined as the number of bits required to
;     load the shift registers internal to the LMX PLL. Organize
;     the Data Streams into Bytes, for efficient transfer via
;     MICROWIRE.
;
;     Single PLL's require sending 2 data streams, 3 bytes each.
;
;     Dual PLL's require sending 4 data streams, 3 bytes each.
;
;
;
;   This example used LMX2320.EXE, available for download from
;     "http://www.national.com", to calculate the appropriate values
;     for the Data Streams.  Please note, that LMX2320.EXE will
;     display the values with the LSB (Least Significant Bit) on the left,
;     and the MSB (Most Significant Bit) on the right.  Transcribe these
;     numbers, with the MSB on the right hand side. Add enough leading
;     zeroes (to the MSB side), so that the number of bits is an integer
;     multiple of 8 (8, 16, 24, 32, etc.).
;
;     Work through the following example to make certain that the
;     modification is clear.  The application note explains the (below)
;     example in further detail.
;
;
;
;   Example #1: LMX2320; Output=1.020 GHz, XTAL=20 MHz, Reference=200 kHz
;               Parameters are:  A=44, N=5100, B=79, R=100, P=64.
;
;   Number of Data Streams = 2
;   Number of Bytes per Stream = 3
;   Data Stream #1 = 004F58         (0000 0000 0100 1111 0101 1000)
;   Data Stream #2 = 0080C9         (0000 0000 1000 0000 1100 1001)
;
;     therefore...

STREAMS:  .BYTE 0x02                    ; Number of Data Streams
BYTESPER: .BYTE 0x03                    ; Bytes per Data Stream
                                        ;
DATA:     .BYTE 0x00, 0x4F, 0x58        ; Data Stream #1
          .BYTE 0x00, 0x80, 0xC9        ; Data Stream #2
```

```
;
;
;
;-------------------------------------------------------------------------
;
;  Subroutines are located below this point...
;
;
;
;  MICROWIRE Data Transmission Subroutine: "uWire"
;
;    The calling routine should have the "data byte" in the
;      Accumulator.  This subroutine will shift this byte
;      via the MICROWIRE port.  Returns to calling program,
;      after all 8 bits have been shifted out
uWire:    LD X, #SIOR           ; Point X register to Serial I/O Register
          X A, [X]              ; Store Accumulator in SIOR
          RBIT 5, PORTGD        ; Force CLOCK=LOW
          SBIT 3, CNTRL         ; Enable MICROWIRE Mode
          SBIT 2, PSW           ; Set BUSY flag, initiating transfer
SETBUSY:  IFBIT 2, PSW          ; Done Yet?
          JMP SETBUSY           ; Keep checking, under BUSY=LOW
          RET                   ; Return to calling program

;
;
;
;-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - -
;  Software Time Delay Loop: "DELAY"
;
;    Two nested loops, count for 65,536 times.  Using
;      the internal R/C in the COP8SA, the DELAY routine
;      will run for about 3/4 of one second.  This
;      is enough time to see an LED blink.

DELAY:                          ; Time Delay Subroutine
                        ;
          LD R2, #0xFF          ; Outer Loop set to 256
PAUSE:    LD R0, #0xFF          ; Inner Loop set to 256
DLOOP:    DRSZ R0               ; Count down Inner Loop
          JP DLOOP              ; Still in the Inner Loop?
          DRSZ R2               ; Count down Outer Loop
          JP PAUSE              ; Still in the Outer Loop?
          RET                   ; Time's up!

;
;
;
;=========================================================================
          .END START
```

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
Americas
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com

www.national.com

National Semiconductor Europe
Fax: +49 (0) 1 80-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 1 80-530 85 85
English Tel: +49 (0) 1 80-532 78 32
Français Tel: +49 (0) 1 80-532 93 58
Italiano Tel: +49 (0) 1 80-534 16 80

National Semiconductor Asia Pacific Customer Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: sea.support@nsc.com

National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507