General Purpose RF Synthesizer Block and Control Software

Per Björklund
Ulf Kylenfall, SM6GXV
Department of Earth and Space Sciences
Chalmers University of Technology
Onsala Space Observatory
SE 439 92 Onsala, Sweden

Introduction

In radio astronomy, there is often a need for various CW RF sources. Often they are realized as readymade instruments tuned to a fixed frequency or smaller built-in blocks that provide LO:s within an instrument. Several manufacturers for these exist, but delivery times and configuration flexibility varies. Often, the cost for custom made units is considerable. It was therefore decided that an in-house knowledge was desired. As a project within a project, a general purpose synthesizer block was designed, built and programmed. This application note describes this block. It will not describe any PLL theory other than the most basic concepts of interfacing.

Concept

A single frequency RF synthesizer consists of basically three parts:

* VCO (external or internal with the synthesizer microcircuit)
* Synthesizer microcircuit
* CPU with control software

Start-up should be automatic. After power-on, the synthesizer module/block should initialize itself, and provide an output signal without any user interaction. In the old days, synthesizers were programmed using parallel interfaces. If a single frequency was desired and no changes ever needed, fixed division ratios could easily be set using jumpers. Today, parallel programming has been replaced with serially programming using a three wire interface LE, CLK and DATA. The synthesizer microcircuit must be loaded with predefined values during initialization. The main project for which the synthesizer block was developed was a heterodyne converter for a water vapor radiometer. A pass band at ~4GHz is down converted to ~1GHz. For this, either a 3.25 or 3.9 GHz synthesizer is used. The final design contains a band switching software, but this is beyond the scope of this article. (Only some details in the CPU schematic shows this).

VCO

Due to a wide range of VCO:s available, Minicircuits was chosen as supplier. Their products are available in either coaxial or surface mount packages. The "Raw" VCO performance in terms of phase noise was found to be adequate for the purpose of this system. Since the synthesizer blocks were designed for fix frequency use, there was no need for wide band versions. The surface mount versions could also be mounted on a PCB using hand tools. For the 3.25 GHz version, Minicircuits ROS3320-219 was chosen.

Synthesizer

As a result of the Cell Phone miniaturization, microcircuit packages have become very small. There are several suitable products on the market, but many of them are available as Chip Scale Packages only. These are almost impossible to mount manually. Eventually, Analog Devices’ ADF4154 was used. It is available in a TSSOP package and can be mounted on a PCB using a solder pen (and microscope).

Software development

Analog Devices offers several tools for PLL development. There is an on-line register calculator, and a comprehensive datasheet, but the methodology and register setting "flow" is not very accurately described. This since the 4154 needs register reconfiguration as part of the initialization process. For the analog parts of the PLL, we used the ADSIM PLL software. It provides all values for the charge pump circuity. For the register configuration, we used the evaluation board software. This software is intended for a PC as evaluation board controller, but the software also provides all register settings very straight forward and convenient. The software packages can be downloaded from Analog Devices home page after registration.

CPU

The task of the CPU is to load the synthesizer with the register settings that will provide proper division ratios and also set some of the values for the charge pump currents and optimizations in terms of low noise or low spur output. Knowledge of the Atmel series of processors made us choose the ATTINY 2313. This processor is available in a plastic
DIP and it can be programmed and reprogrammed using either an ICP or using a separate development system, in our case the STK500. Atmel offers free download of the AVR Studio, and this was used for source code development.

**Synthesizer & VCO Schematic**

*Page 19*

**Pin and Signal Description:** 10 MHz Reference Clock (10 MHz) LE Latch Enable. Low during synthesizer programming. CLK Clock. Synchronous with serial data programming. DATA Serial data input. LOCK Analog Lock Detect Output. -SYNT_EN Logic LOW to turn on VCO (Not used in single channel configuration) +3 and +5 Volts are also required.

The schematic is provided as a reference only. U3 is an isolation amplifier that prevents the clock from leaking backwards so that internal spurs does not occur in the output spectrum. It also provides an adequate RF signal level for the RFIN_A pin on the PLL microcircuit. The MUX pin on the CPU has to be programmed for LOCK/UNLOCK output using register configuration. This is however not present in the software listing below.

**IF Block CPU Schematic**

*Page 20*

**Pin and Signal Description:** The schematic of the CPU has two groups of PLL control signals. Since the Water vapor radiometer heterodyne mixer required two frequencies and since there were no VCO that could cover both frequencies, it was decided to use two synthesizer units and a RF switch. For single frequency use; simply omit all CH2 wiring. The device was programmed using "Internal Clock" option which makes use of an internal RC oscillator. There is no need for a quartz oscillator for the CPU.

**Software listing**

The reader will discover that some texts are commented out. In order to keep maximum flexibility, the comments were kept so that future debugging could be easily performed by just activating the assembler statements.

**Software listing**

```assembly
; Pers variabel: ; include "C:\AVR\include\tn2313def.inc"
ulfs variabel: ; include "C:\AVR\AvrAssembler2\Appnotes\tn2313def.inc"

;-------------------------------------------
; Definitioner
;
; def temp =r16
; def temp2 =r17
; def temp3 =r18
; def Count_Bit =r19;Which bit is being sent
; def Count =r20;How many characters is to be sent
; def UpCount =r21;How many characters has been sent?
; (For Wait function)
; def Byte =r22
; def Output =r23

; PORTB bit 0 LE
; 1 DATA
; 2 CLK

;*************************************************
; Interupt vectors
; org 0x00
rjmp Reset

;*************************************************
Reset:
; Ini stack
ldi temp,high(RAMEND)
out sph,temp
ldi temp,low(RAMEND)
out spl,temp

; PORTB Output
ldi temp,$7
out DDRB,temp

; Set Output default
ldi Output,0
out PORTB,Output
clr UpCount ; Clear up-counter
sei
rjmp preMain

;*************************************************
Main1:
rcall Wait_10mS
rcall Wait_10mS
rcall Wait_10mS
ldi temp,$4
out PORTB,temp
rcall Wait_10mS Skall vara kvar?
clr temp
out PORTB,temp
```
; rjmp Main

; TEST  
; -------------------------------------
preMain:
rcall Disable_LE
; rcall Wait_10mS
; -------------------------------------

Main:

; Wait for loop filter to settle
; Commented out: Not necessary for 4154
; ==========================
; rcall Wait_10mS

; Main Initialization
; ====================

: R3 - Clear Test Bits (00 00 03 - Never changes)
; =====
rcall Enable_LE
ldi Byte,$00
rcall Send
ldi Byte,$00
rcall Send
ldi Byte,$03
rcall Send
rcall Disable_LE

: R3 - Select Low Noise Mode (3C7 - Never changes)
; =====
rcall Enable_LE
ldi Byte,$00
rcall Send
ldi Byte,$03
rcall Send
ldi Byte,$C7
rcall Send
rcall Disable_LE

: R2 - Enable Counter Reset during initialization
; =====
rcall Enable_LE
ldi Byte,$00
rcall Send
ldi Byte,$0C
rcall Send
ldi Byte,$46
rcall Send
rcall Disable_LE

: R1 - R-Divider Register
; =====
rcall Enable_LE
ldi Byte,$04
rcall Send
ldi Byte,$40
rcall Send
ldi Byte,$51
rcall Send
rcall Disable_LE

: R0 - N-Divider Register
; =====
rcall Enable_LE
ldi Byte,$28
rcall Send
ldi Byte,$80
rcall Send
ldi Byte,$00
rcall Send
rcall Disable_LE

: R2 - Disable Counter Reset: Start Synthesizer
: with loaded values
; =====
rcall Enable_LE
ldi Byte,$00
rcall Send
ldi Byte,$0C
rcall Send
ldi Byte,$42
rcall Send
rcall Disable_LE
End:
rjmp End

;-------------------------------

; SUBROUTINES

Send:

ldi count_bit,$8
Send_

mov temp,Byte
;/Clk_up
cpi count_bit,0 ;Which bit is being sent?
breq Stop_Send
dec count_Bit
andi temp,$80
sbrs temp,7
rjmp Send_Low
rcall Send_High

Left_Shift:

isl Byte
rjmp Send_

Stop_Send:
ret

;-------------------------------
Conclusion

A general purpose RF synthesizer block has been described. The reader should be able to find information in this application note that will assist in the design of similar units using the resources listed.

Resources used

The schematic drawings and the circuit board layout was designed using Capture CIS and Allegro XL from Cadence Design systems, Inc.

The software was created using AVR Studio and programmed into the processor using STK500 from ATMEL Corp.

References

www.minicircuits.com
www.analog.com
www.atmel.com
www.avrfreaks.com
www.cadence.com

@