



Digital Tuned FM Stereo Radio  
LV24000 / LV24001 / LV24002

# APPLICATION NOTE

## AN2400S02

PRELIMINARY

---

**REVISION HISTORY**

- V0.0** 09-March-2004  
- Initial version
- V0.1** 26-March-2004  
- Changed IF S-curve phase
- V0.2** 04-May-2004  
- Added flowcharts

**TABLE OF CONTENTS**

INTRODUCTION .....	4
HOST HARDWARE REQUIREMENTS.....	5
BASIC INFORMATION.....	6
Radio regions.....	6
Display VS tuned frequency.....	6
BASIC ROUTINES .....	7
Writing data to the LV2400x.....	7
Reading data from the LV2400x .....	7
Measure frequency .....	7
With software timing window.....	7
With external clock (using CLK_IN).....	7
BASIC THEORY .....	8
Redefine the LV2400x registers for software ease .....	8
Dealing with negative DAC control registers .....	8
Dealing with the 7½ bits FM_CAP register .....	8
Finding out the value for a DAC control register .....	9
Quick setting RF frequency.....	10
IMPLEMENTING RADIO FUNCTIONS.....	12
Initialization .....	12
Hardware initialization .....	12
Software initialization .....	12
Setting frequency .....	12
Scan radio station .....	13
APPENDIX A: NOTATIONS USED BY PSEUDO-CODE .....	14
APPENDIX B: BIG STEP TUNING.....	15
APPENDIX C: FINE STEP TUNING .....	15
APPENDIX D: INITIALIZE THE QUICK SET FREQUENCY DATA.....	16
APPENDIX E: CALCULATE CAP/OSC VALUE.....	17
APPENDIX F: QUICK SET RF-FREQUENCY .....	18
APPENDIX G: SCAN RADIO STATION .....	19
APPENDIX H: MISCELLANEOUS ROUTINES .....	21
Constant A for RF-frequency coefficient .....	21
CalculateCoeff .....	21
InterpolateX.....	21
InterpolateY.....	21
APPENDIX G: FLOW CHARTS .....	22

## INTRODUCTION

The Sanyo LV2400x family of products introduces a new concept of FM radio IC's that offer great functionality combined with excellent performance and without the use of external components. To achieve these characteristics new principles of tuning and control of the FM radio have been developed. This application note is intended to give insight in the principles that give the LV2400x product its compelling specifications.

Typical tuners IC's use an external clock and a PLL to generate the tuned RF frequency. In the LV2400x series, software on the host takes care of tuning the IC. This results in a very fast tuning and maximum flexibility in controlling and configuring the IC: scanning the whole FM band in less than 2 seconds by weak reception (no station), setting a FM frequency within 150 ms<sup>1</sup>.

For tuning to a radio station three oscillator frequencies are important: The RF frequency, the IF frequency and the stereo decoder frequency. All these frequencies are programmed by software by adjusting four predefined setting frequencies registers: a DAC control register for each frequency (fine-tuning) and a capacitor switch control (coarse-tuning) for the RF frequency due to its great frequency range.

The above mentioned registers need to be set to specific values in order to correctly receive a radio station. The mechanism used for this is a "set and measure" algorithm using smart interpolation to quickly find out the right register's value.

Measuring the frequency is done by counting pulses ( $p$ ) generated by the oscillator (RF, IF or stereo decoder) during a known time ( $t$ ). The actual frequency can then be easily determined by dividing the ( $p$ ) by ( $t$ ). If the required frequency is not correct, the frequency register is reprogrammed. The new value can be estimated accurately by knowing the behavior of the oscillator when changing settings. This way only a few steps are required to find the right setting. Once the right frequency is found it can be locked by internal AFC logic so drift due to temperature or voltage changes is avoided.

The following chapters of the application note go in more detail on how to control the LV2400x. For a quick overview of the software refer to APPENDIX G: FLOW CHARTS.

---

<sup>1</sup> Based on 8-bit microprocessor with instruction clock at 2 MHz, driving 3-wires bus with 3 GPIO pins.

## HOST HARDWARE REQUIREMENTS

Access to the LV2400x IC is done through the 3-wire bus:

CLOCK	Data strobe, input to the LV2400x
NR_W	Command (Write or read data), input to the LV2400x
DATA	Bi-directional pin: input to the LV2400x when NR_W is high, output from the LV2400x when NR_W is low.

Therefore, the host CPU should supply 3 GPIO pins, configured as following:

CLOCK	Output of the host
NR_W	Output of the host
DATA	Bi-directional: Output of the host when NR_W is high (write to LV2400x mode). Input of the host when NR_W is low (read from LV2400x mode). <u>Optional requirement of DATA-line:</u> <i>Possibility to generate interrupt.</i>

To reduce the software load during measuring of a frequency, a (8-bit/16-bit) timer is preferable. The timer generates interrupt when it rolls over and continues with counting until the control software disables it.

When the timer is not available, the control software has to perform a busy waiting loop of  $\pm 32$  ms for frequency measurements.

When an external clock, which is supplied by the host hardware, is connected to CLK\_IN pin of the LV2400x, the busy waiting loop can be reduced (depends on the CLK\_IN frequency).

Also, to reduce the software load on the host CPU, the DATA pin can be configured as an interrupt pin after the LV2400x is appropriately set up. In this manner, the control software doesn't have to poll the LV2400x for radio events (like stereo/mono mode changing, signal strength drops, measuring with CLK\_IN done...)

---

## BASIC INFORMATION

### Radio regions

The popular radio regions are summarized below:

Region	Band limit	De-emphasis
Japan	76 MHz - 90 MHz	50 $\mu$ s
Europe	87.5 MHz – 108 MHz	50 $\mu$ s
USA	87.5 MHz – 108 MHz	75 $\mu$ s

### Display VS tuned frequency

Radio stations are marked by their displayed frequency. To receive a radio station, a radio receiver must be tuned so that it can lock the IF for demodulation.

For the LV2400x, the tuned frequency is the sum of the displayed frequency and the preset IF frequency, in formula:

$$\text{Tuned FM frequency} = \text{displayed frequency} + \text{preset IF frequency}$$

For example: when the IF frequency of LV2400x is preset at 110 kHz, it must be tuned at 88.51 MHz to receive the radio station at 88.4 MHz.

## **BASIC ROUTINES**

### **Writing data to the LV2400x**

Refer to the data sheet of the actual LV2400x for the format of the data pattern

### **Reading data from the LV2400x**

Refer to the data sheet of the actual LV2400x for the format of the data pattern

### **Measure frequency**

#### *With software timing window*

##### Using host 's timer

TBD

##### Using "busy waiting"

TBD

#### *With external clock (using CLK\_IN)*

##### Using interrupt from LV2400x

TBD

##### Polling for counting done event

TBD

## BASIC THEORY

### Redefine the LV2400x registers for software ease

#### ***Dealing with negative DAC control registers***

The LV2400x has some DAC control registers in negative logic (i.e. TBD, TBD). This means when increasing the content of those registers, the frequency is decreased.

The FM\_OSC (TBD) register is on the contrary, positive logic. Software can use the following conversion to convert the negative DAC control registers to positive logic:

Logical value (SoftwareValue) to physical value (HardwareValue) conversion:

$$\text{SoftwareValue} = 255 - \text{HardwareValue}$$

Physical value (HardwareValue) to logical value (SoftwareValue) conversion:

$$\text{HardwareValue} = 255 - \text{SoftwareValue}$$

Applying the conversion on the negative DAC control registers, software can work with the logical value to have all the DAC control registers of LV2400x in positive logic. A low-level routine converts the logical value to physical value before writing it to the hardware.

Having all DAC control registers in positive logic makes it possible for the software to use just one algorithm for determining the DAC control register value by a frequency.

#### ***Dealing with the 7½ bits FM\_CAP register***

Working with the FM\_CAP register requires some care because:

- It's in negative logic.
- The value range is not continuous due to the overlapping of bit 7 and bit 6 of this register (combination 01b and 10b have the same range)

Software can apply following conversion to the FM\_CAP:

Logical value (SoftwareValue) to physical value (HardwareValue) conversion:

```

If (SoftwareValue <64)
    HardwareValue = 255 - SoftwareValue
Else
    HardwareValue = 255 - 64 - SoftwareValue
  
```

Physical value (HardwareValue) to logical value (SoftwareValue) conversion:

```

If (HardwareValue <128)
    SoftwareValue = 255 - 64 - HardwareValue
Else
    SoftwareValue = 255 - HardwareValue
  
```

After the conversion, software will get a linear FM\_CAP register in positive logic. The logical value range is from 0 to 191.

### **Finding out the value for a DAC control register**

Tuning the LV2400x to a frequency is finding out which value should be programmed in the companion DAC control register.

There are 3 tunable frequencies on the LV2400x:

- IF frequency: controlled by the IF\_OSC register.
- Stereo decoder clock: controlled by the SD\_OSC register.
- RF frequency: controlled by the FM\_CAP (coarse step) and FM\_OSC (fine step) register.

These 4 registers can be tuned with 1 algorithm (see APPENDIX B: BIG STEP TUNING). The fine step tuning (see APPENDIX C: FINE STEP TUNING) can be optionally used when precision is needed.

During radio station scanning, the FM\_CAP and FM\_OSC must be adjusted to vary the RF frequency. So the values for these 2 registers should be quickly determined for a quick scan. Software can approach them via their math model, as described in next session.

**Quick setting RF frequency**

Setting the RF frequency is finding out the values for FM\_CAP and FM\_OSC register. The FM\_CAP adjusts the RF frequency in big steps. The FM\_OSC adjusts the RF frequency in small steps.

The RF frequency is generated with formula:

$$\omega^2 = 1/(LC) \quad \text{where} \quad \omega = 2\pi f \quad (f \text{ is the desired RF})$$

L is the inductance of the coil  
C is the capacitance (controlled by FM\_CAP/FM\_OSC)

Or

$$C = 1/(L\omega^2)$$

$$C = 1/(L \cdot 4\pi^2) \cdot 1/f^2$$

The first part of the equation (1/(L4π²)), is a coefficient which varies with L but L can be considered as constant for a specific LV2400x in a specific hardware design. So when we redefine this part as a constant A (see further APPENDIX H: MISCELLANEOUS ROUTINES - Constant A for RF-frequency coefficient), a coefficient for a frequency f can be calculated as follows:

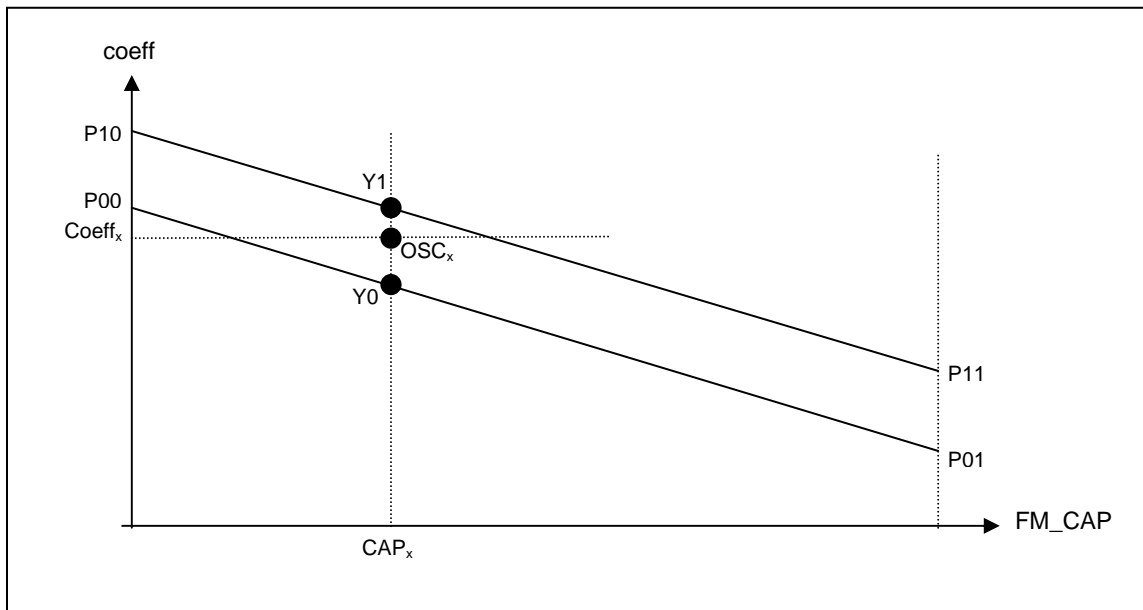
$$\text{Coeff} = (A/f^2)$$

When we use the coefficient to approach the RF frequency, we will have a linear characteristic so that a FM\_CAP value for a RF frequency can be interpolated between the point FM\_CAP=low and the point FM\_CAP=high.

When we apply the FM\_OSC to the FM\_CAP, we will have 4 points, divide into 2 sets as follows:

FM_OSC	FM_CAP	Point
Low	Low	P00 (set 0)
Low	High	P01 (set 0)
High	Low	P10 (set 1)
High	High	P11 (set 1)

In graphic:



---

With these 4 points, the FM\_CAP and FM\_OSC of a RF-frequency  $f_x$  can be calculated as follows:

- Calculate  $\text{coef}_x$  from RF frequency  $f_x$  ( remember that  $\text{coef}_x = A/f_x^2$  )
- Using set 0, value  $\text{CAP}_x$  can be interpolated.
- With  $\text{CAP}_x$ , value Y0 and Y1 can be interpolated from set 0 and set 1
- The FM\_OSC value  $\text{OSC}_x$  can be interpolated from Y0 and Y1.

The calculated value  $\text{OSC}_x$  needs to be corrected because we approach the RF frequency with linear characteristic. The value of  $\text{OSC}_x$  can be adjusted as follows:

- Write  $\text{CAP}_x$ ,  $\text{OSC}_x$  to the LV2400x
- Measure the RF frequency  $f_m$  at this point
- Calculate the  $\text{coef}_m$  from  $f_m$
- Determine the correction factor ( $\text{coef}_x - \text{coef}_m$ )
- Apply the correction factor to Y0 and Y1
- Interpolate the  $\text{OSC}_x$  value again from the corrected Y0 and Y1

So to set RF frequency, we only need 1 measurement (if precision is required)  
See APPENDIX E: CALCULATE CAP/OSC VALUE for pseudo code.

The 4 point P00, P01, P10, P11 can be measured once when the software is initialized  
(See APPENDIX D: INITIALIZE THE QUICK SET FREQUENCY DATA)

---

## IMPLEMENTING RADIO FUNCTIONS

### Initialization

#### *Hardware initialization*

- Write default registers' values to the LV2400x.  
Refer to the data sheet of the actual LV2400x for registers and registers' values
- Preset the IF-frequency (initialize the IF DAC control)  
Refer to the data sheet of the actual LV2400x for the default value of the IF-frequency  
It is recommended to use both big step tuning and fine step tuning algorithm
- Preset the stereo decoder clock (initialize the stereo decoder DAC control). This step can be postponed until the stereo mode is enabled (saving start up time)  
Refer to the data sheet of the actual LV2400x for the default value of the stereo clock.  
It is recommended to use both big step tuning and fine step tuning algorithm

#### *Software initialization*

- Initialize the global software data  
See APPENDIX D: INITIALIZE THE QUICK SET FREQUENCY
- Setting the radio region: initialize the FM band limit for scanning, set the de-emphasis
- Restore last user's settings like last radio station, stereo/mono... (this step is optional)

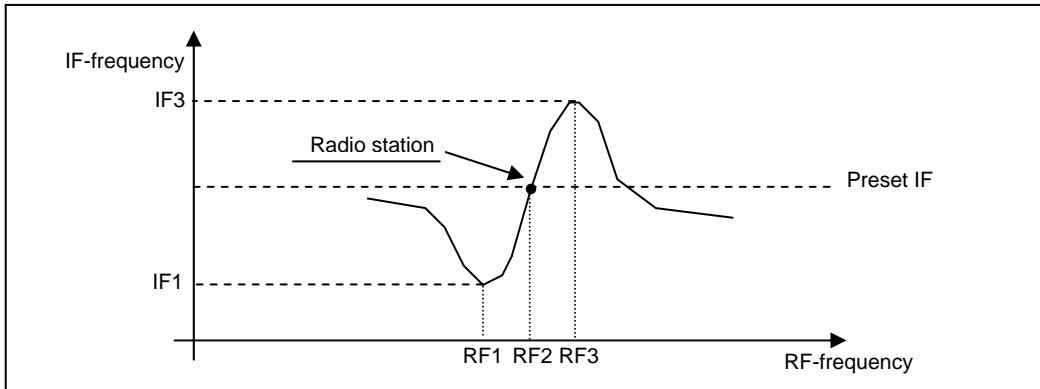
HINT: Audio can be muted during initialization.

### Setting frequency

- Calculate the RF frequency from the displayed frequency
- Disable AFC
- Mute the audio to hide noises during tuning process
- Enable measure mode of LV2400x (clear EN\_MEAS# bit of Control Register A)
- Select OS\_FM\_OSC as output (OUTPUT\_SELECT bits in control register A)
- Calculate the FM\_CAP and FM\_OSC (see APPENDIX E: CALCULATE CAP/OSC VALUE)
- Restore the output select of control register A
- Restore the measure mode of LV2400x
- Restore the audio mute state
- Restore AFC state

**Scan radio station**

A radio station is characterized by the S-curve of IF-frequency (see picture below): when the RF-frequency is at F1, the IF-frequency is lower than the preset IF, and when the RF-frequency is at F3, the IF-frequency is higher than the preset IF. The difference RF-IF when RF varies from F1 to F3 remains the same: that is the displayed frequency of the radio station. The range F3-F1 depends on the strength of the radio station.



Scanning for a radio station is done in 2 phases:

1. Detecting the IF edge with sufficient delta IF (the swing between IF1 and IF3): Vary the RF-frequency until 2 points with sufficient IF-swing are measured.
2. Validating the IF edge with the knowledge that increasing RF will decrease IF (and vice versa), and the equation  $RF1 - IF1 = RF2 - IF2$  (=displayed frequency) must be valid for a RF-frequency can be concluded as a radio station.

See APPENDIX G: SCAN RADIO STATION for complete scan description.

**APPENDIX A: NOTATIONS USED BY PSEUDO-CODE**

//	<i>Comment</i>
=	<i>Assignment</i>
=	<i>Test for equal</i>
!=	<i>Test for not equal</i>
>	<i>Test for greater</i>
<	<i>Test for smaller</i>
>>	<i>Shift right (bit wise)</i>
<<	<i>Shift left (bit wise)</i>
abs	<i>absolute function (example: abs(1-2)=1)</i>

## APPENDIX B: BIG STEP TUNING

Input:

*x1*: Low value to write to the DAC control register (first point of the interpolating)  
*x2*: high value to write the DAC control register (Second point of the interpolating)  
*f*: frequency to be set

Write *x1* to the target DAC control register

*f1* = Measure frequency at *x1*

Loop

Write *x2* to the target DAC control register

*f2* = Measure frequency at *x2*

Check *f2* against *f* – terminate loop if *f2* within margin

*step* = InterpolatingX(*f*, *x1*, *x2*, *f1*, *f2*)

if *step* is 0 terminate loop // can not approach the frequency with interpolating

*x\_new* = *x1* + *step* // Caution: *step* can be negative!

Move point2 to point1 by assigning *x1*=*x2*; *f1*=*f2*

Make new point2 by assigning *x2* = *x\_new* // *f2* will be measured

Repeat loop

NOTE:

- The interpolating routine *interpolatingX* is described in APPENDIX H: MISCELLANEOUS ROUTINES
- For LV2400x: the logical value 10 can be used for *x1*, 240 for *x2*
- Write to DAC control register: the algorithm works with logical values. Apply conversion to physical value if necessary.

## APPENDIX C: FINE STEP TUNING

Input:

Initial step: any value > 0

Register Value: Start value of a DAC control register for the fine tuning

*f*: frequency to be set

*step* = initial step

Loop

Register Value = Register Value + *step* // *step* can be negative!

Write Register Value to the target DAC control register

*f1* = Measure Frequency at Register Value

Check *f1* against *f* – terminate loop if *f1* within margin

If *f1*<*f*

*step* = absolute(*step*) // increase register value

set TOO\_SMALL flag

if *step* == 1 also set APPROACH\_UP\_1 flag

If *f1*>*f*

*step* = -absolute(*step*) // decrease register value

set TOO\_BIG flag

if *step* = -1 also set APPROACH\_DOWN\_1 flag

If both APPROACH\_UP\_1 and APPROACH\_DOWN\_1 flags set

terminate loop // approached with *step*=1: best fit value found

else if both TOO\_BIG and TOO\_SMALL flags set

*step* = *step*/2

If *step*==0 make *step*=1

repeat loop

NOTE:

- Apply this routine when precision is required (getting the best fit value for a DAC control register)
- Initial Step can be set as 16 for LV2400x
- Write to DAC control register: the algorithm works with logical values. Apply conversion to physical value if necessary.

---

**APPENDIX D: INITIALIZE THE QUICK SET FREQUENCY DATA**

*SwOscLow = 10*  
*SwOscHigh = 240*

*SwCapLow = 0*  
*SwCapHigh = 191*

*Write SwOscLow to the LV2400x*

*Write SwCapLow to the LV2400x*  
*Measure f00*  
*Coef00 = CalculateCoeff(f00)*

*Write SwCapHigh to the LV2400x*  
*Measure f01*  
*Coef01 = CalculateCoeff(f01)*  
*Write SwOscHigh to the LV2400x*

*Write SwCapLow to the LV2400x*  
*Measure f10*  
*Coef10 = CalculateCoeff(f10)*

*Write SwCapHigh to the LV2400x*  
*Measure f11*  
*Coef11 = CalculateCoeff(f11)*

**NOTE:**

- *Values Coef00, Coef01, Coef10, Coef11 should be stored for later usage*
- *The software CAP/OSC values are logical values. Apply conversion to physical values before writing them to the LV2400x.*

## APPENDIX E: CALCULATE CAP/OSC VALUE

Input: the RF-frequency  $f$

PrecisionLevel: NONE, LOW, MEDIUM, HIGH

Output: the calculated CAP, OSC value

// Determine correction action

if (PrecisionLevel is NONE)

    MeasureTime = 0ms

    CorFreq = 0Hz

if (PrecisionLevel is LOW) OR (PrecisionLevel is MEDIUM)

    MeasureTime = 32ms

    CorFreq = 32Hz

if (PrecisionLevel is HIGH)

    MeasureTime = 64ms

    CorFreq = 16Hz

CorFreq = CorFreq \* DividerFactor // For RF –frequency; DividerFactor=256

Coef = CalculateCoeff( $f$ )

CapValue = InterpolateX(Coef, SwCapLow, swCapHigh, Coef00, Coef01)

Write SwOscLow to LV2400x

Done = FALSE

While (Done is FALSE)

    CoefLo = InterpolateY(CapValue, SwCapLow, swCapHigh, Coef00, Coef01)

    CoefHi = InterpolateY(CapValue, SwCapLow, swCapHigh, Coef10, Coef11)

    CoefRange = CoefLo – CoefHi

    Write CapValue to the LV2400x

    If MeasureTime is not 0

        Fcur = Measure RF-frequency with MeasureTime

        CoefFcur = CalculateCoeff(Fcur)

        CoefCor = CalculateCoeff(Fcur + CorFreq)

        CoefLo = CoefFcur + CoefCor

        CoefHi = CoefCur – CoefRange - CoefCor

    else

        CoefCur = CoefLo

    if Coef is in range [CoefLo, CoefHi]

        OscValue = InterpolateX(Coef, SwOscLow, SwOscHigh, CoefLo, CoefHi)

        If OscValue is in range [swOscLow, SwOscHi] Done = TRUE

    if (Done is FALSE)

        CapNew = InterpolateX(Coef, CapValue, SwCapHigh, CoefCur, Coef01)

        If CapNew is equal to CapValue

            if Coef is smaller than CoefCur

                CapValue = CapValue + 1

            Else

                CapValue = CapValue – 1

        Else

            CapValue = CapNew

        Repeat the while-loop

### NOTE:

- The software CAP/OSC values are logical value. Apply conversion to physical values before writing them to the LV2400x.
- Interpolating is described in APPENDIX H: MISCELLANEOUS ROUTINES

## APPENDIX F: QUICK SET RF-FREQUENCY

Input: the RF-frequency  $f$   
 PrecisionLevel: NONE, LOW, MEDIUM, HIGH

Calculate CAP/OSC value for frequency  $f$  (see APPENDIX E: CALCULATE CAP/OSC VALUE)  
 If (PrecisionLevel is NONE) OR (PrecisionLevel is LOW)  
   Write OscValue to LV2400x  
   Exit

if (PrecisionLevel is MEDIUM)  
   ValidateCapOsc with 8ms measurement  
   Exit

if (PrecisionLevel is HIGH)  
   ValidateCapOsc with 64ms measurement  
   Exit

### ValidateCapOsc

Input: the RF-frequency  $f$   
 MeasureTime

Retry = 0  
 Loop

  Write CapValue to LV2400x  
   BigStepTuning (see APPENDIX B: BIG STEP TUNING)  
   If FAILED  
     if  $f < \text{Current RF}$   
       CapValue = CapValue + 1  
     Else  
       CapValue = CapValue – 1  
     Retry = Retry + 1  
     If Retry is greater than 3  
       Exit with unreachable frequency failure  
     Else  
       Repeat loop  
   if OK  
     Done, exit loop

## APPENDIX G: SCAN RADIO STATION

Input: The field strength level of the desired station *UserFs*  
 The scan direction: *iDir*  
     +1: scan up (increase RF-frequency)  
     -1: scan down (decrease RF-frequency)

Adjust the *UserFs*/Calculate *ScanStep*, *IfSwing*

<i>UserFs</i>	<i>QuickFsLevel</i>	<i>ScanStep</i>	<i>IfSwing</i>
0	0	60 kHz	40 kHz
1	0	65 kHz	45 kHz
2	0	70 kHz	50 kHz
3	1	75 kHz	55 kHz
4	2	75 kHz	55 kHz
5	3	75 kHz	55 kHz
6	4	75 kHz	55 kHz
7	5	75 kHz	55 kHz

### Scan algorithm:

```

If Current RF is a valid station
    Step = 100 kHz           // Step away from current station
Else
    Step = 0
IfSwing = abs(IfSwing) * iDir // Apply scan direction as sign in IfSwing
IfSwing = IfSwing * -1       // Negate IfSwing because of IF phase
PrecisionLevel = LOW
CheckFS = TRUE
IF1 = 0
Rf = Current RF-frequency
ValidStation = FALSE

While ValidStation is FALSE
    Rf = Rf + (Step * iDir)
    Step = ScanStep // Always use scan step after the first iteration
    If Check RF limit failed: exit with limit failure
    QuickSetFrequency(Rf, PrecisionLevel)
    If FAILED exit with unreachable frequency failure
    Update display if necessary
    If CheckFS is TRUE
        if MeasuredFS is smaller than QuickFsLevel
            PrecisionLevel = NONE
            Repeat the While-loop

        if MeasuredFS is greater or equal to QuickFsLevel
            RfCur = Measure RF with 8ms
            If RfCur is behind Rf
                PrecisionLevel = NONE
                Repeat the while-loop
        else
            CheckFS = FALSE

    PrecisionLevel = LOW
    IF2 = Measure IF-frequency with 8ms
    EdgeFound = FALSE

    If IF1 is not 0
        DeltaIF = IF1 - IF2
        If (IfSwing is greater than 0) AND (DeltaIF is greater than IfSwing)
            EdgeFound = TRUE
        If (IfSwing is smaller than 0) AND (DeltaIF is smaller than IfSwing)
    
```

---

*EdgeFound = TRUE*

*IF1 = IF2 // Take over the IF for next time*  
*If EdgeFound is FALSE*  
     *Repeat the while-loop*

*// Correct edge: Pre-check the field strength*  
*if MeasuredFS is smaller than QuickFsLevel*  
     *Repeat the while-loop*

*// Pinpoint the radio station*  
*ValidStation = FindFmStation*  
*If ValidStation is FALSE*  
     *CheckFS = TRUE*  
     *Repeat the while-loop*

*// Post-check field strength*  
*if MeasuredFS is smaller than UserFS*  
     *ValidStation = FALSE*  
     *Repeat the while-loop*

*// Hereafter: exit while-loop with ValidStation is TRUE: station is found at current RF*

### **FindFmStation**

*IfFreq = Measure IF-frequency with 32ms*  
*If IF within range [Preset IF  $\pm$  15kHz] exit with StationFound is TRUE*  
*RfFreq = Measure RF-frequency with 32ms*  
*Retry = 0*

*Loop*

*RfStep = Preset IF - IfFreq // Caution: RfStep is signed and can be negative*

*RfFreq = RfFreq + RfStep*

*QuickSetFrequency to set RfFreq with precision is HIGH*

*If FAILED exit with StationFound is FALSE*

*IfFreq = Measure IF-frequency with 32ms*

*If IF within range Preset IF  $\pm$  15kHz*  
         *exit with StationFound is TRUE*

*else*

*Retry = Retry + 1*

*If Retry reaches value 4*

*Exit with StationFound is FALSE*

*Repeat loop*

### **NOTE:**

- Quick set frequency is described in APPENDIX F: QUICK SET RF-FREQUENCY

## APPENDIX H: MISCELLANEOUS ROUTINES

### Constant A for RF-frequency coefficient

When working with floating point is possible, the coefficient can be calculated as it is i.e.  $1/f^2$ .

But when it is difficult to work with floating point, constant A must be so chosen that it shifts the coefficients to integer values that are big enough to cover the frequency range 66 MHz-116 MHz without overflow. The resolution of the coefficient must be about 4 kHz to work with the RF-frequencies

Consider the following suggestion:

Derive from formula  $\text{coef} = A/f^2$

$$\text{Coef} = \frac{A}{f^2} = \frac{\frac{A1}{f} * A2}{f} * A3$$

When A1, A2 and A3 are chosen as follows

$$A1 = 2^{32} - 1$$

$$A2 = 2^{16}$$

$$A3 = 2^8$$

The coefficients will fit in 32 bits integer.

### CalculateCoeff

Input: The RF-frequency  $f$  in Hz

Output:

$$f_{\text{kHz}} = f/1000$$

if ( $f_{\text{kHz}} == 0$ )

$$\text{Coef} = 0$$

else

$$\text{Coef} = ((A1 / f) * A2) / f * A3$$

### InterpolateX

Input: ExpectedY,  $x1$ ,  $x2$ ,  $y1$ ,  $y2$

Output:

If  $y1 == y2$

$$X = 0$$

else

$$X = (\text{ExpectedY} - y1) * (x2 - x1) / (y2 - y1) + x1$$

### InterpolateY

Input: ExpectedX,  $x1$ ,  $x2$ ,  $y1$ ,  $y2$

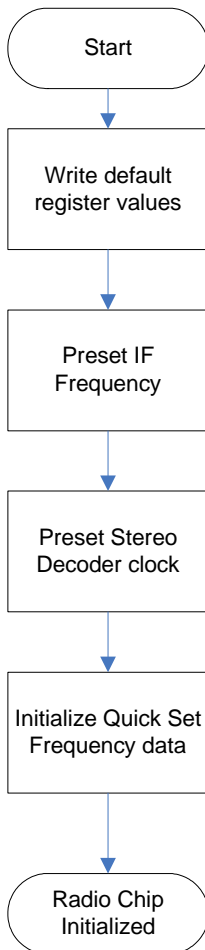
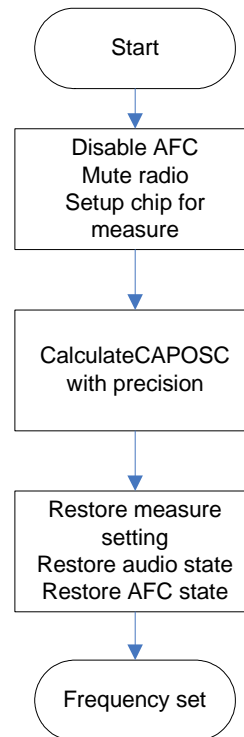
Output:

If  $x1 == x2$

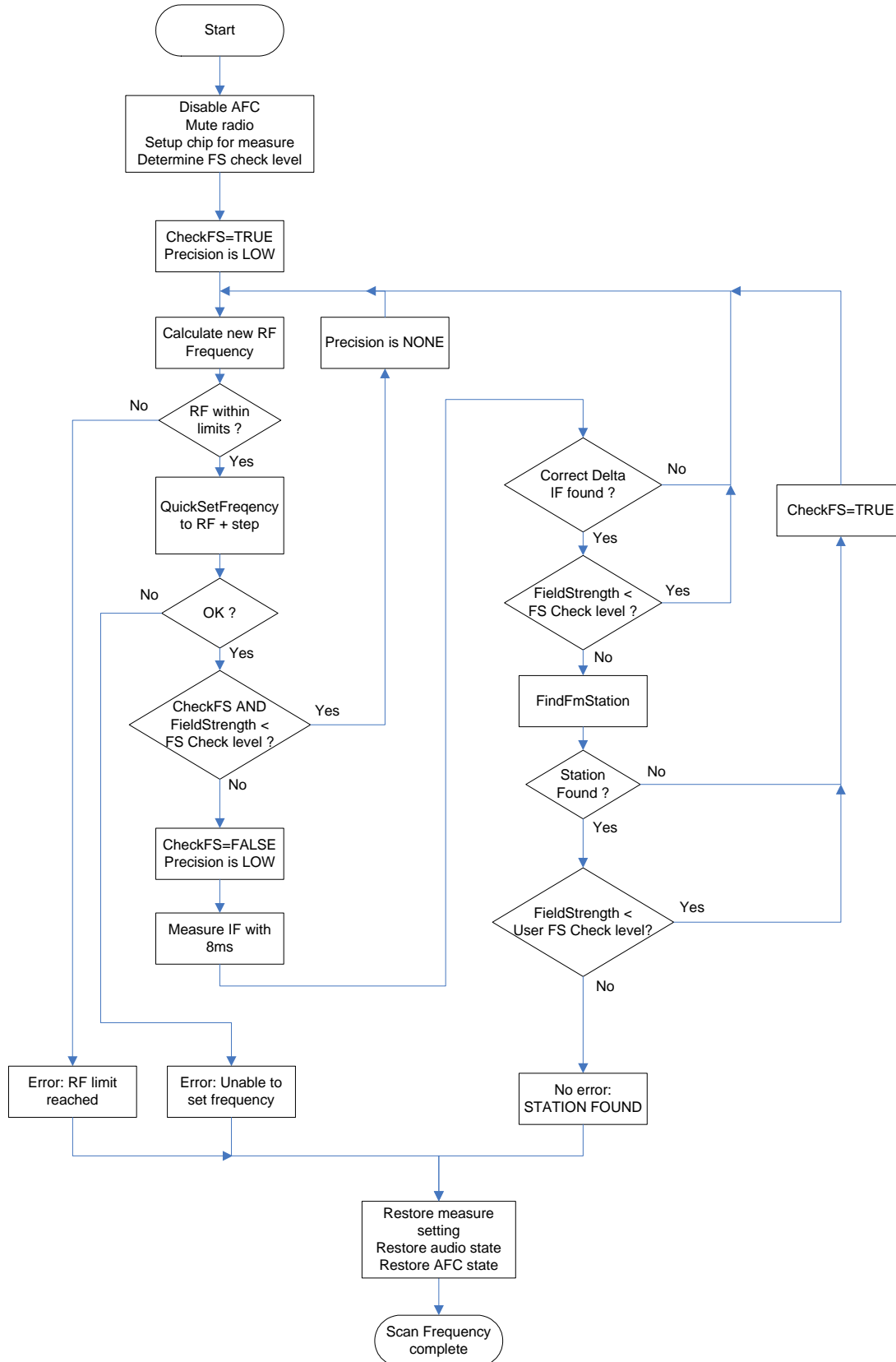
$$Y = 0$$

else

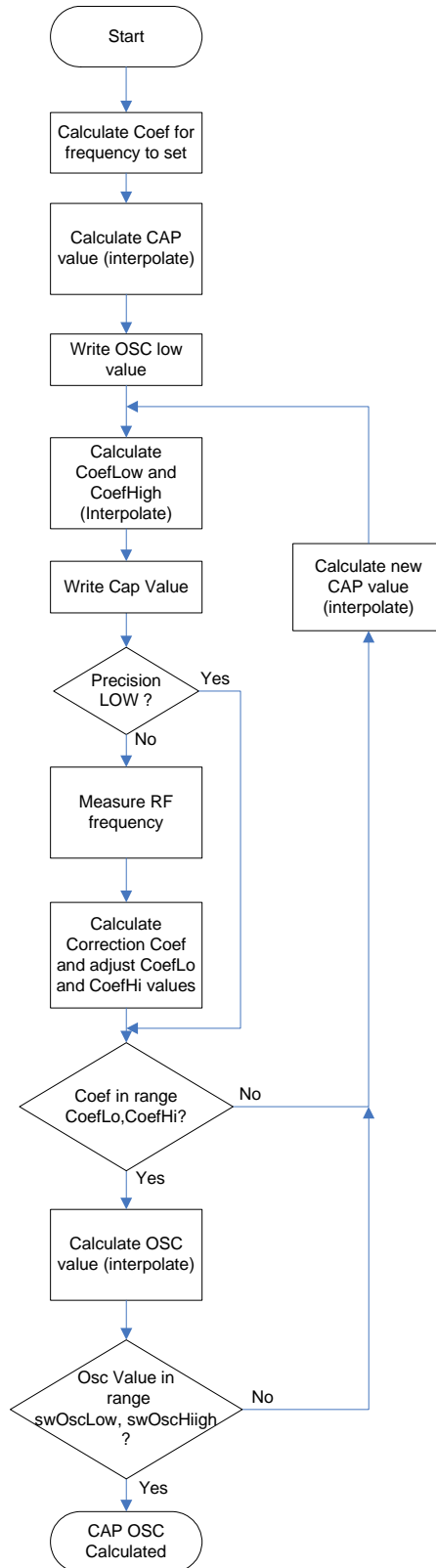
$$Y = (\text{ExpectedX} - x1) * (y2 - y1) / (x2 - x1) + y1$$

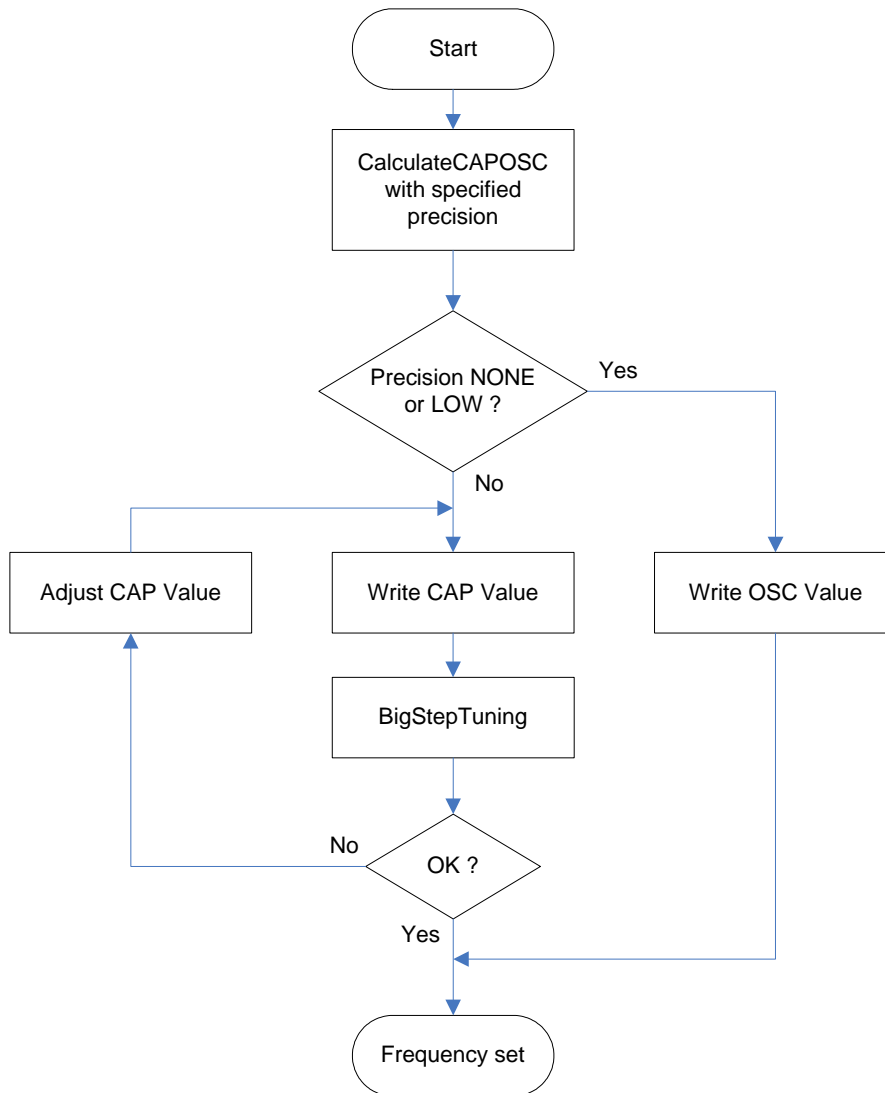
**APPENDIX G: FLOW CHARTS****Initialize Radio****Set Frequency**

**Scan Frequency**

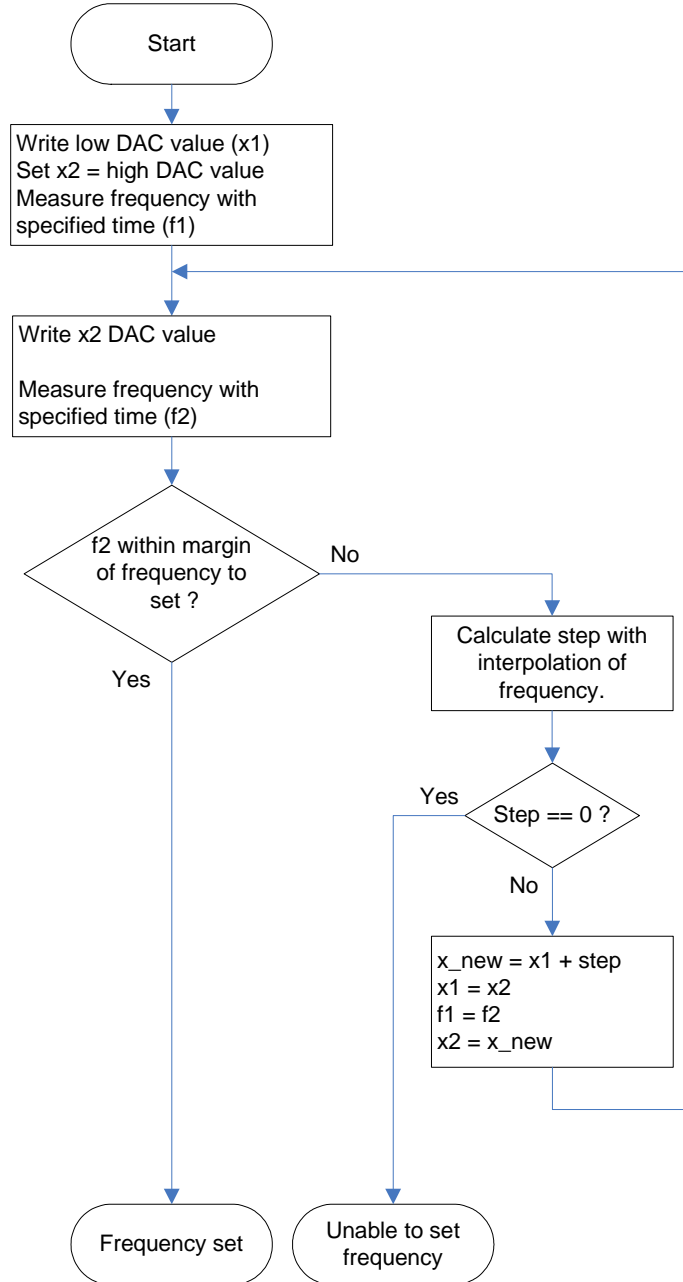


Calculate CAP OSC

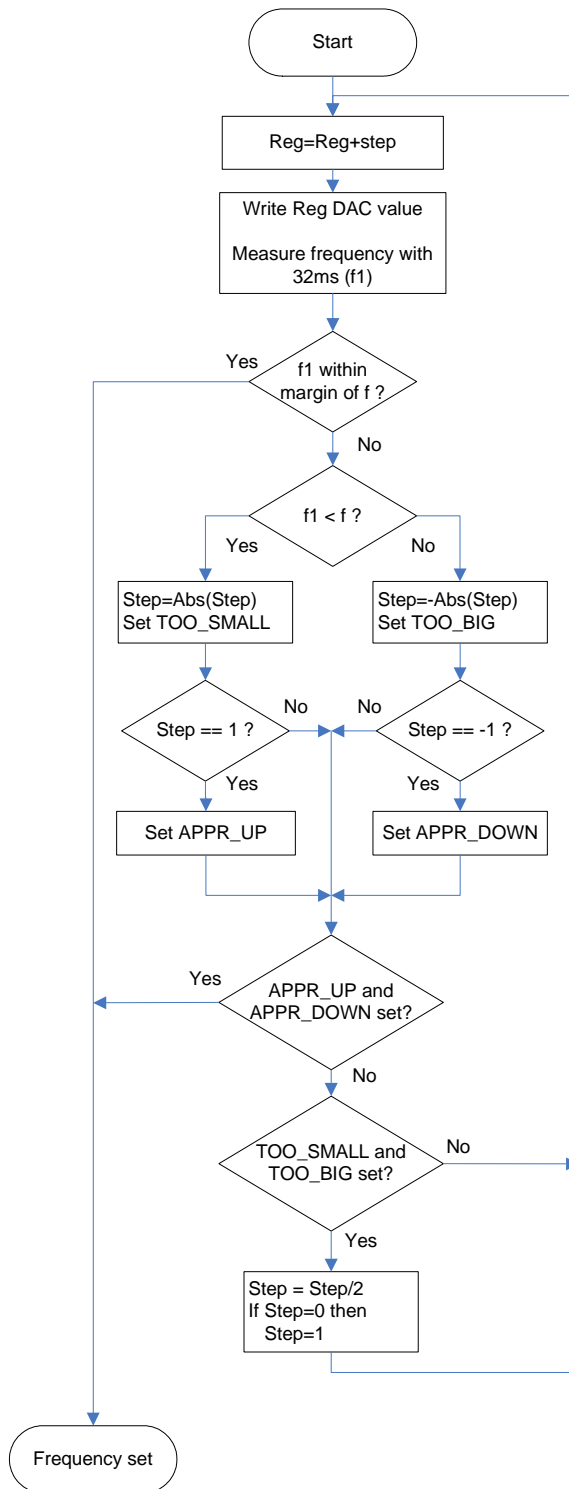


**QuickSetFrequency**

**BigStepTuning**



**FineStepTuning**



**FindFmStation**

