QN8006 Driver SDK API Programming Guide

Sep, 2009

Contents

REVISION HISTORY

REVISION	CHANGE DESCRIPTION	DATE
0.01	Draft	04/15/09
0.1	Modify driver API examples	05/11/09
0.2	Remove some diagrams	09/15/09

STATEMENT:

Users are responsible for compliance with local regulatory requirements for low power unlicensed FM broadcast operation. Quintic is not responsible for any violations resulting from user's intentional or unintentional breach of regulatory requirements in personal or commercial use.

1 Overview

1.1 Driver API Architecture

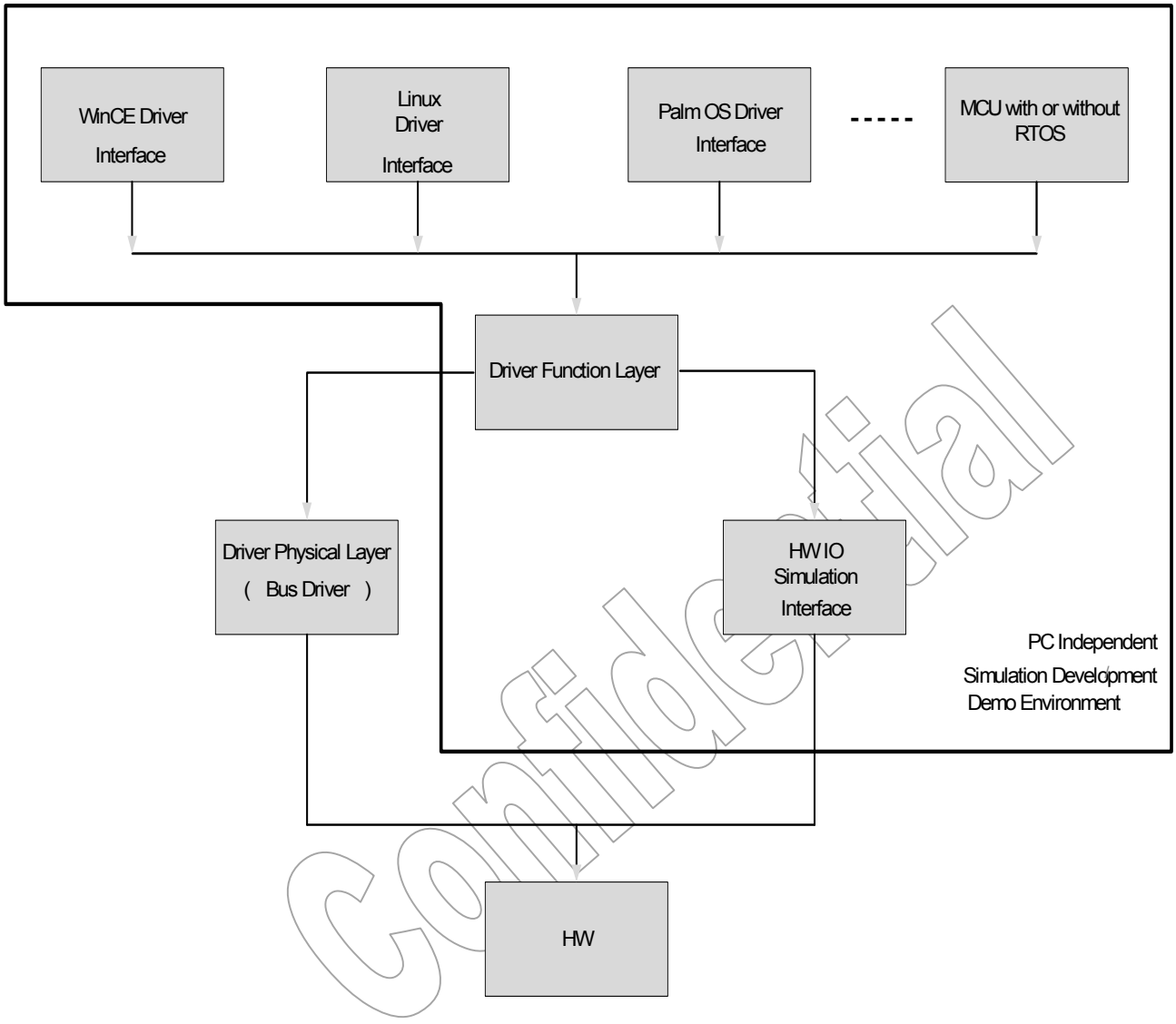


Figure 1: QN8006 SDK Driver API Architecture

1.2 Driver API RX Operation

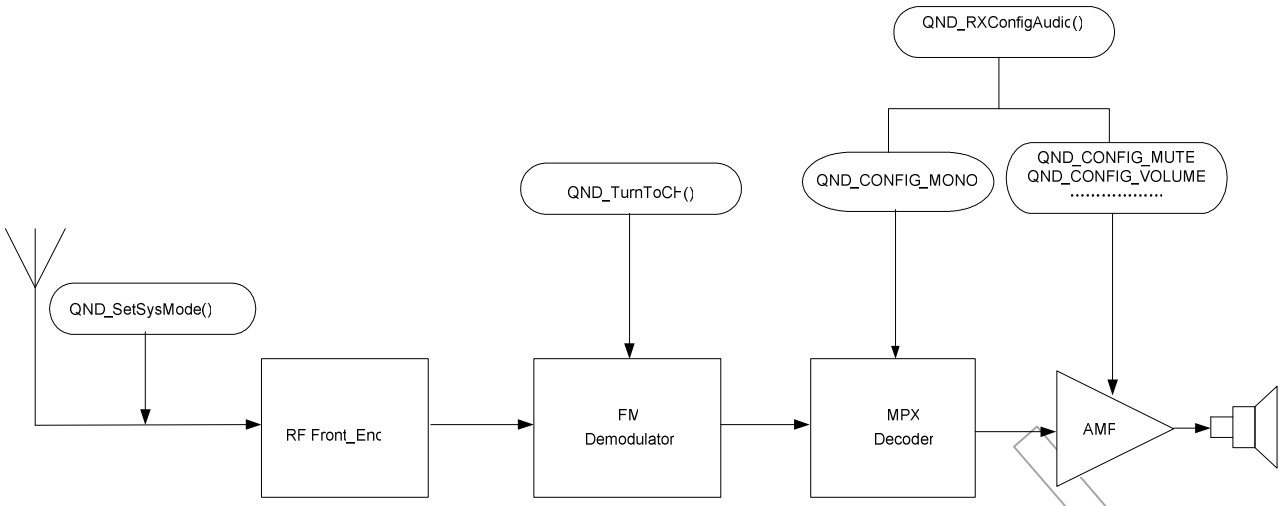


Figure 2: QN8006 SDK Driver API RX Operation

1.3 Driver API TX Operation

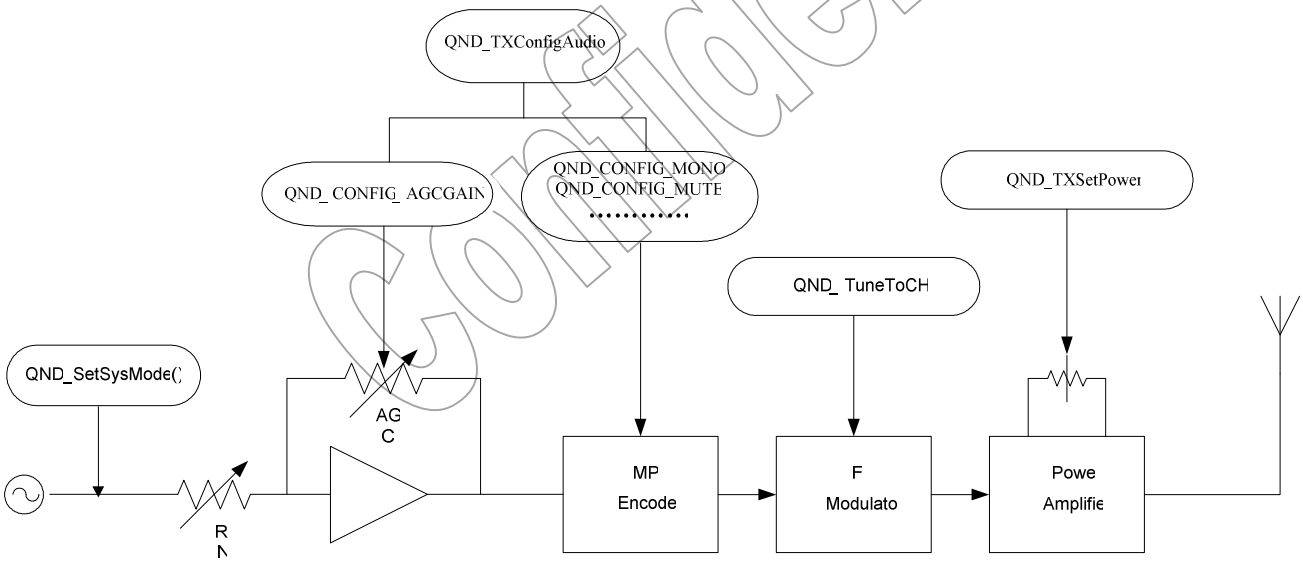


Figure 3: QN8016 SDK Driver API TX Operation

1.4 Driver API CCA Operation

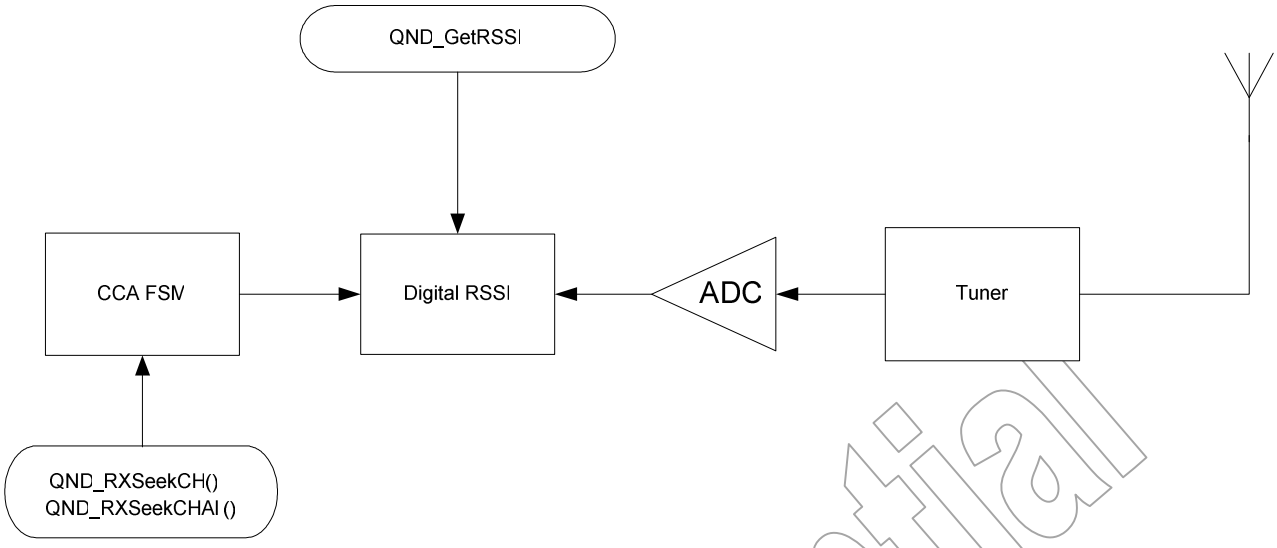


Figure 4: QN8006 SDK Driver API CCA Operation

2 API Descriptions

2.1 QND_Delay

```
*********************************************************************
int QND_Delay(UINT16 ms)
*********************************************************************
```

Description: Delay for some ms

Parameters:

ms: ms counts

Return Value:

None

Note: Before using this function, you need set delay unit into one millisecond. Both TX and RX need.

Flowchart: omit

Note: Both TX and RX need this function.

This function should be reimplemented depending on hardware and system used.

For example, in Windows System this function can be implemented as:

```
void QND_Delay(UINT16 ms)

{

    Sleep(ms);

}
```

In ARM or MCS-51 System ,this function can be implemented as:

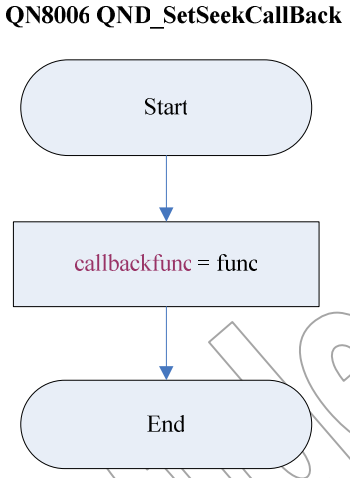
```
void QND_Delay(UINT16 ms)

{

    UINT16 i,k;

    for(i=0; i<3000;i++) {

                for(k=0; k<ms; k++) {}

    }

}
```

Of course the 3000 should be modified depending on the hardware speed. If the system supports multitask and need to switch between these tasks, the QND_Delay function should be reimplemented with the system's internel delay function.

2.2 QND_SetSeekCallBack

```
**********************************************************************
void QND_SetSeekCallBack(QND_SeekCallBack func)
**********************************************************************
Description: set call back function which can be called between seeking
             channel
Parameters:
     func : the function will be called between seeking
Return Value:
     None
```

Flowchart:



Note:

If you want use QND_RXSeekCH function or QND_RXSeekCHAll function, you can call this function to set CallBack function which can help you to know all valid channels during seeking channel.

Example:

```
Void MySeekCallback(UINT16 ch,UINT8 bandtype)
{
     if(bandtype== BAND_FM)
          printf("Found a valid channel:%.2f MHZ\n",ch/100.0);
     else
          printf("Found a valid channel:%d KHZ\n",(int)ch);

}
Void TestSeekChannel()
{
   UINT8 nCh;
    QND_init();
    QND_SetSysMode(QND_MODE_FM| QND_MODE_RX);
    QND_SetSeekCallBack(MySeekCallback);
    nCh=QND_RXSeekCHAll(7600,10800, QND_FMSTEP_100KHZ,6,1);
```

```
    if(nCh>0)
    {
      printf("%d channel(s) found\n",(int)nCh);
    }
    else
    {
      printf("No channel found\n");
    }
}
```

2.3 QND_Init

```
*********************************************************************
int QND_Init()
*********************************************************************
```

Description: Initialize device to make it ready to have all functionality ready for use.

Parameters:

None

Return Value:

1: Device is ready to use.

0: Device is not ready to serve function.

Flowchart:

2.4 QND_GetRSSI

```
*********************************************************************
UINT8 QND_GetRSSI(UINT16 ch)
*********************************************************************
```

Description:Get the RSSI value, call QNF_SetCh to set specify CH

Parameters:

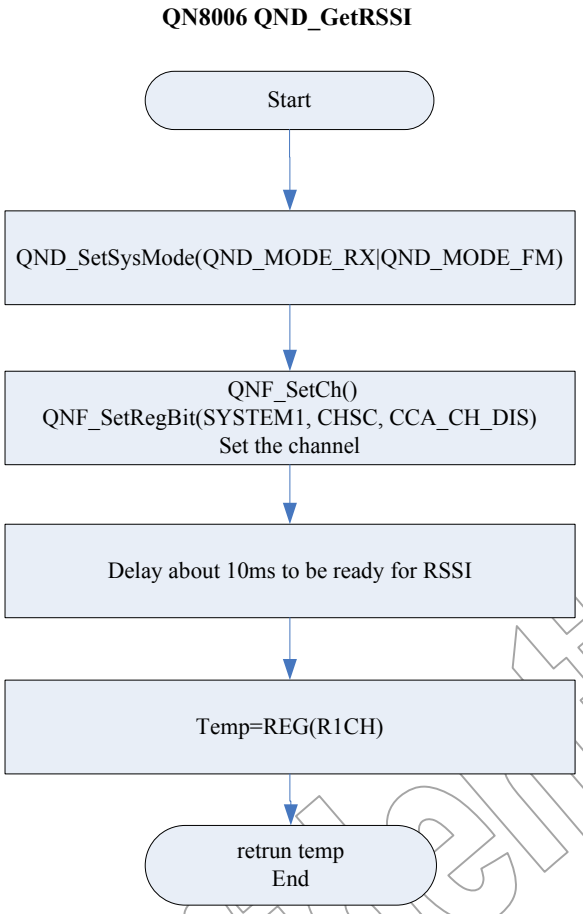
Ch

to be set channel which the RSSI will be get

Return Value:

RSSI value of the channel setted

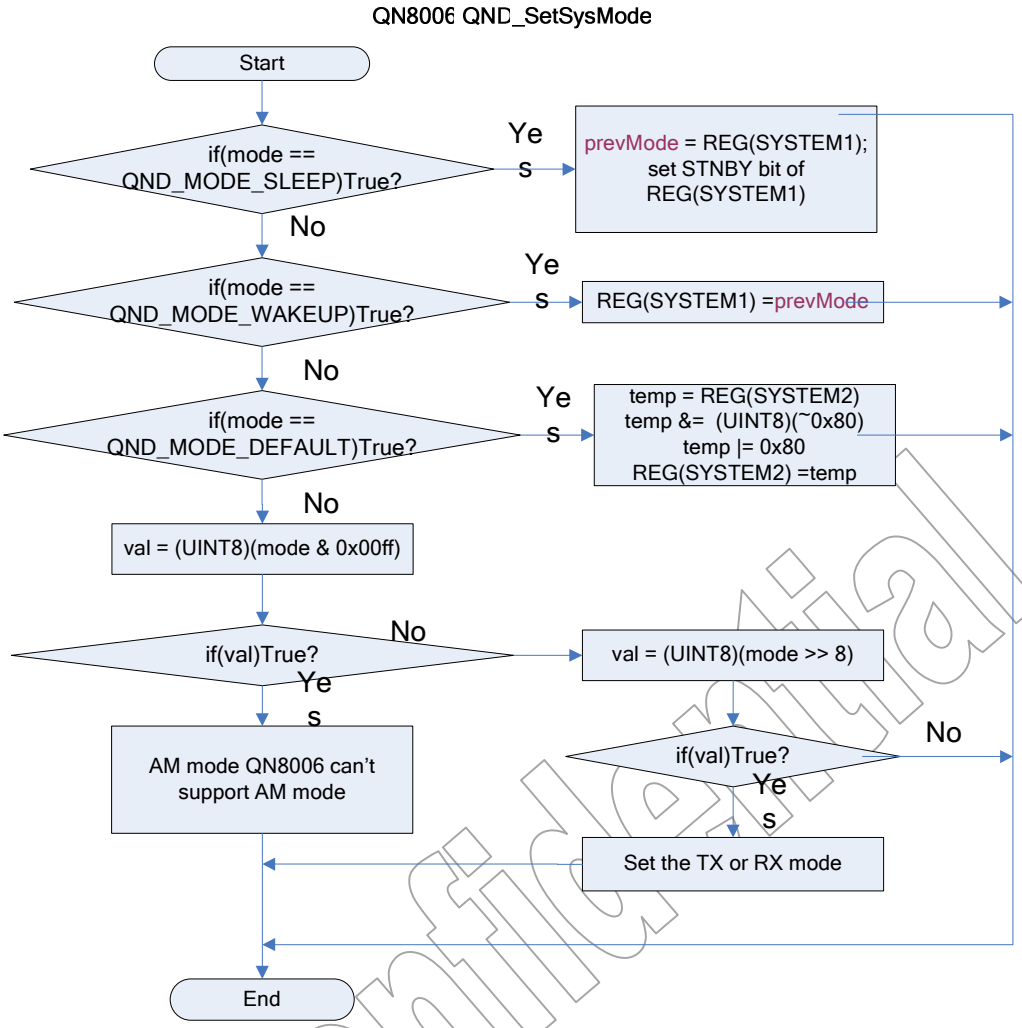
Flowchart:



2.5 QND_SetSysMode

```
*********************************************************************
void QND_SetSysMode(UINT16 mode)
*********************************************************************
Description: Set device system mode (like: sleep, wakeup etc)
Parameters:
    Set the system mode, it will be set by some macro define usually:
     QND_MODE_SLEEP:  set device work on sleep mode
     QND_MODE_WAKEUP: wakeup device
     QND_MODE_RX|QND_MODE_FM:  set device work on FM, RX mode
     QND_MODE_TX|QND_MODE_FM:  set device work on FM, TX mode
     QND_MODE_DEFAULT: Reset to default values
Return Value:
    None
```

Flowchart:



2.6 QND_TuneToCH

```
*********************************************************************
void QND_TuneToCH(UINT16 ch)
*********************************************************************
```

Description: Tune to the specified channel. Make sure the QND_SetSysMode has been called before use this function. If system is working on FM&TX mode, it will turn FM to ch, and start Transmit; if system is working on FM&RX mode, it will turn FM to ch, and start FM play

Parameters:

ch

Set the freqency (unit is 10kHz) to be tuned,

Eg: 101.30MHz will be set to 10130.

Return Value:

None

Flowchart:

2.7 QND_RXConfigAudio

```
*********************************************************************
void QND_RXConfigAudio(UINT8 optiontype, UINT8 option)
*********************************************************************
Description: Config the Rx Audio performance like volume adjust,
             equalizer set, mute set:
Parameters:
optiontype: It indicates what you want to config, it can use some macro
            define to control:eg: VOLUMECONFIG, EQUALIZERCONFIG,
            MUTECONFIG, MONOCONFIG,OUTPUTFORMATCONFIG
option:
set the config value;
if (optiontype == QND_CONFIG_MUTE); 'option' will control muteable,
     0:mutedisable,1:mute enable

if (optiontype == QND_CONFIG_MONO); 'option' will control mono,
     0: QND_AUDIO_STEREO,1: QND_AUDIO_STEREO


Return Value:
     none
```

Flowchart:

2.8 QND_RXSeekCHAll

```
**********************************************************************
UINT8 QND_RXSeekCHAll(UINT16 start, UINT16 stop, UINT16 step, UINT8 db,
UINT8 up)
**********************************************************************
```

Description: Automatically scans the complete FM band and detects all the available channels. A threshold value needs to be passed in for the channel detection.

Parameters:

start
Set the frequency (unit is 10kHz) where scan will be started,
eg: 76.00MHz will be set to 7600.

stop
Set the frequency (unit is 10kHz) where scan will be stopped,
eg: 108.00MHz will be set to 10800.

Step
FM:
QND_FMSTEP_100KHZ: set leap step to 100kHz
QND_FMSTEP_200KHZ: set leap step to 200kHz
QND_FMSTEP_50KHZ: set leap step to 50kHz

db
Set signal noise ratio for channel to be searched.

up:
Set the seach direction :
Up;0,seach from stop to start
Up:1 seach from start to stop

Return Value:
The channel count found by this function
0: no channel found

Flowchart:

2.9 QND_RXSeekCH

```
**********************************************************************
UINT16 QND_RXSeekCH(UINT16 start, UINT16 stop, UINT16 step, UINT8 db,
UINT8 up);
**********************************************************************
```

Description: Automatically scans the frequency range, and detects the first channel.A threshold value needs to be passed in for channel detection.

Parameters:

start
Set the frequency (unit is 10 kHz) where scan will be started,
eg: 76.00MHz will be set to 7600.

stop
Set the frequency (unit is 10kHz) where scan will be stopped,
eg: 108.00MHz will be set to 10800.

step
FM:
QND_FMSTEP_100KHZ: set leap step to 100 kHz
QND_FMSTEP_200KHZ: set leap step to 200 kHz
QND_FMSTEP_50KHZ: set leap step to 50 kHz

db:

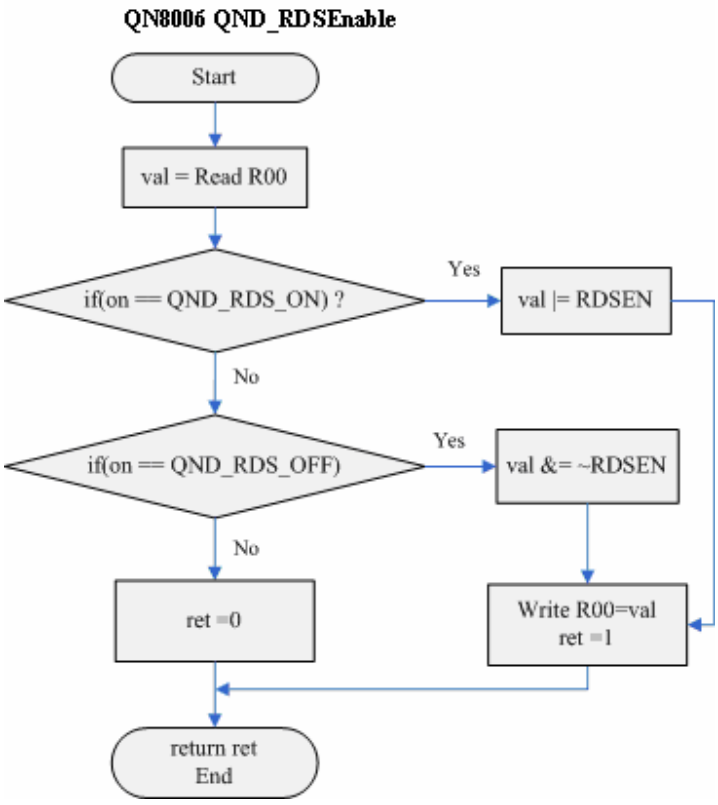
```
      Set threshold for quality of channel to be searched.
up:
      Set the search direction:
            Up; 0, search from stop to start
            Up: 1, search from start to stop
Return Value:
  The channel frequency (unit: 10 kHz)
  0: no channel found
```

Flowchart:

2.10 QND_RDSEnable

```
*********************************************************************
UINT8 QND_RDSEnable(UINT8 on)
*********************************************************************
Description: Enable or disable chip to work with RDS related functions.
Parameters:
          on:
             QND_RDS_ON:  Enable chip to receive/transmit RDS data.
             QND_RDS_OFF: Disable chip to receive/transmit RDS data.
Return Value:
    1: function executed
    0: function didn't execute
```

Flowchart:



2.11 QND_ RDSDetectSignal

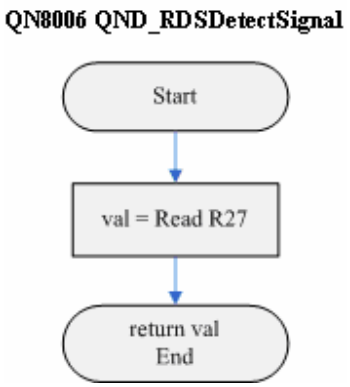
```
*********************************************************************
UINT8 QND_RDSDetectSignal(void)
*********************************************************************
```

Description: To detect RDS status.

Return Value:

Register value of R27.

Flowchart:

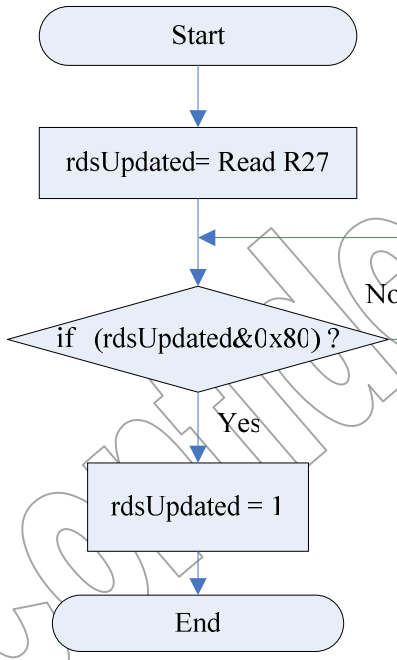


2.12 QND_RDSCheckBufferReady

```
**********************************************************************
UINT8 QND_RDSCheckBufferReady (void)
**********************************************************************
Description:  Check chip RDS buffer status before doing unload of RDS
              data.

Parameters:
    None
Return Value:
    QND_RDS_BUFFER_NOT_READY: RDS buffer is not ready to use.
    QND_RDS_BUFFER_READY:     RDS buffer is ready to use. You can now
                              unload (for RX) data from RDS buffer
```

Flowchart:

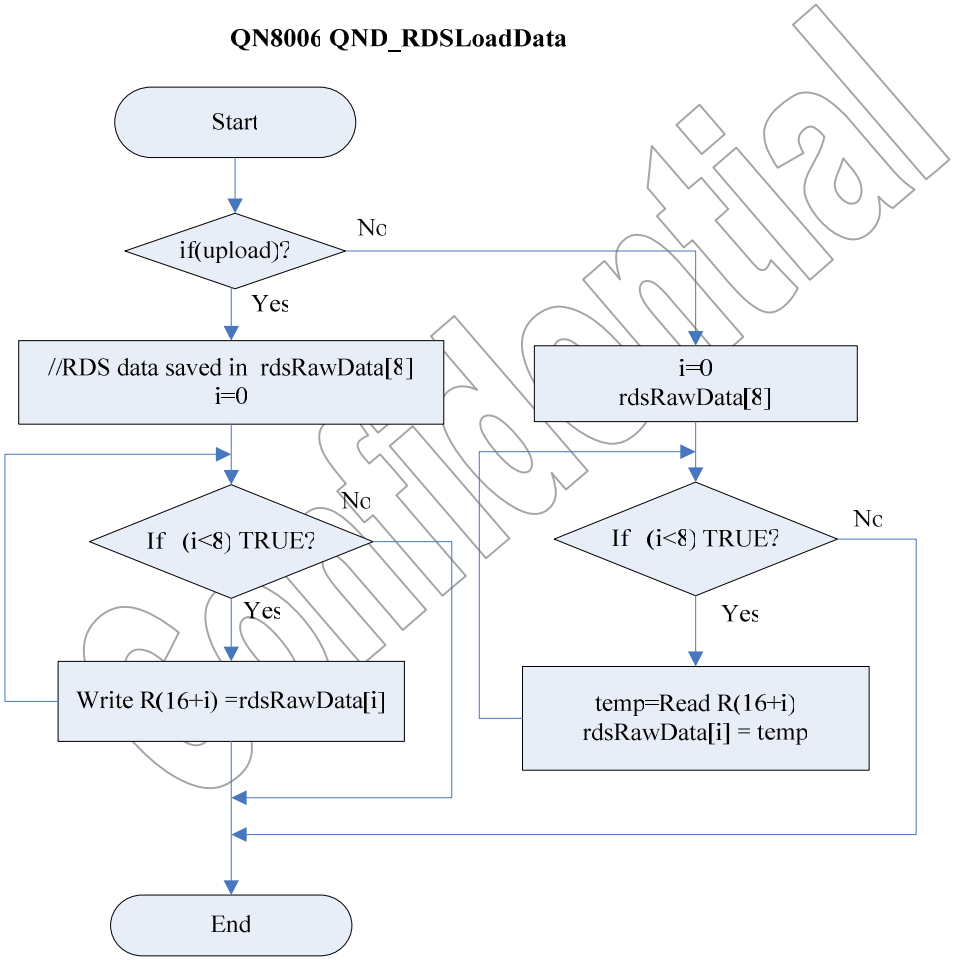
QN8006 QND_RDSCheckBufferReady



2.13 QND_RDSLoadData

```
*********************************************************************
UINT8   QND_RDSLoadData(UINT8 *rdsRawData, UINT8 upload)
*********************************************************************
Description:  Unload (RX) RDS data from on-chip RDS buffer. Before
              calling this function, always make sure to call the
              QND_RDSBufferReady function to check that the RDS is
              capable to unload RDS data.
Parameters:
           rdsRawData : 8 bytes data buffer to unload (on RX mode)
from chip RDS buffer.
        Upload:   0--download
Return Value:
    QND_SUCCESS: function executed
```

Flowchart:



2.14 QND_SetCountry

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

void QND_SetCountry(UINT8 country)

****

Description: Set start, stop, step for RX and TX based on different country.

Parameters:

country: Set the chip used in specified country:

CHINA:

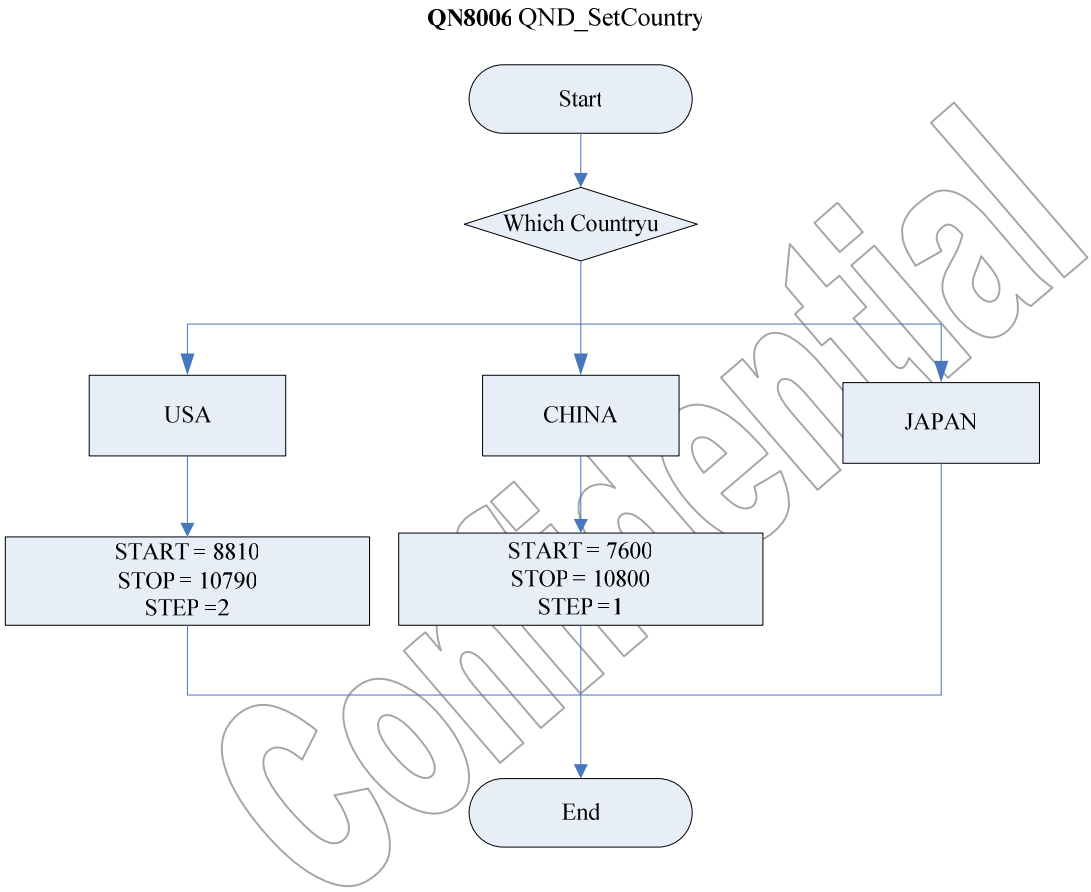
USA:

JAPAN:

Return Value:

None

Flowchart:



2.15 QND_ConfigFMModule

****

Void QND_ConfigFMModule(UINT8 optiontype, UINT8 option)

****

Description: Config the FM modulation setting.

Parameters:

optiontype:

QND_CONFIG_AUDIOPEAK_DEV : audio peak deviation
QND_CONFIG_PILOT_DEV: pilot deviation
QND_CONFIG_RDS_DEV: rds deviation

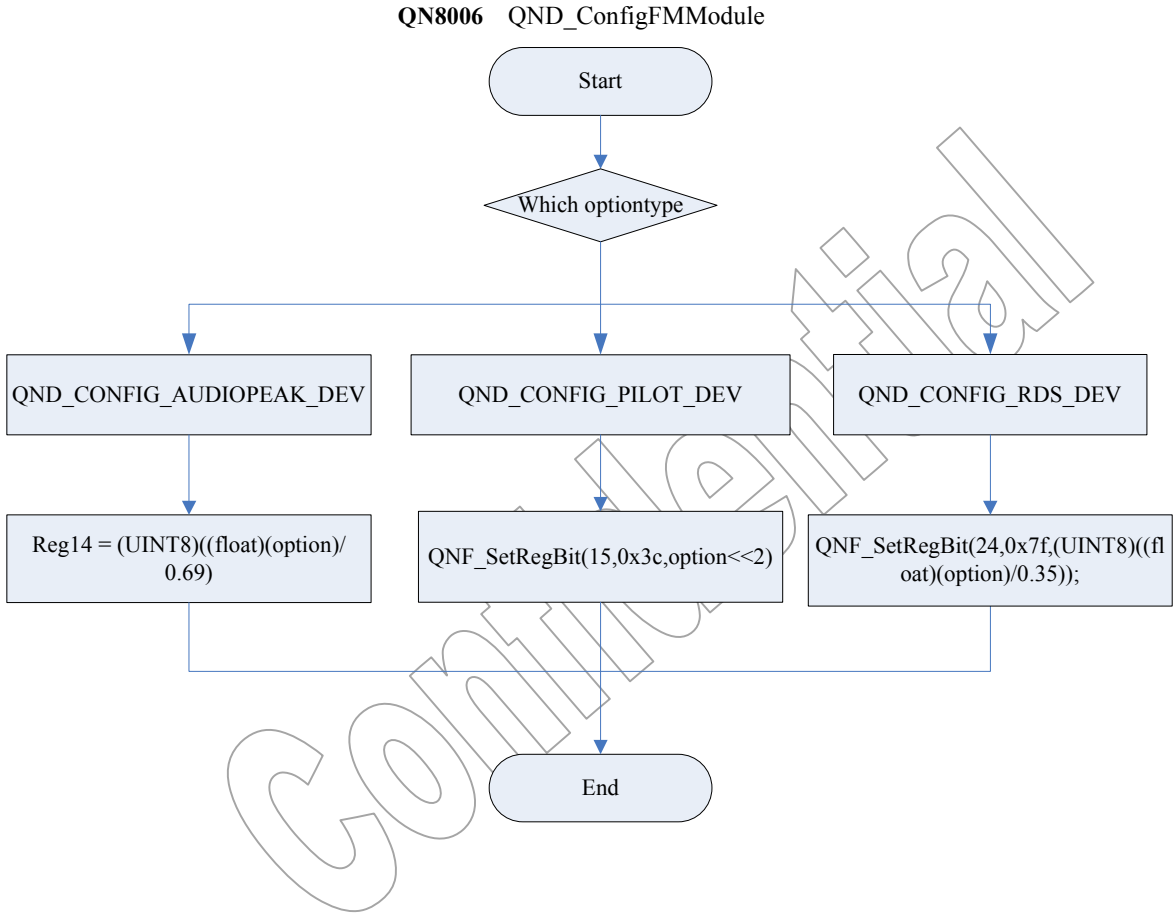
option:

QND_CONFIG_AUDIOPEAK_DEV: 0~165khz
QND_CONFIG_PILOT_DEV: 8~10
QND_CONFIG_RDS_DEV: 1~7.5khz

Return Value:

None

Flowchart:



2.16 QND_LoadDefaultSetting

****

void QND_LoadDefalutSetting(UINT8 country)

****

Description: load some default load some default setting for a certain country load some default setting for a certain country setting for a certain country.

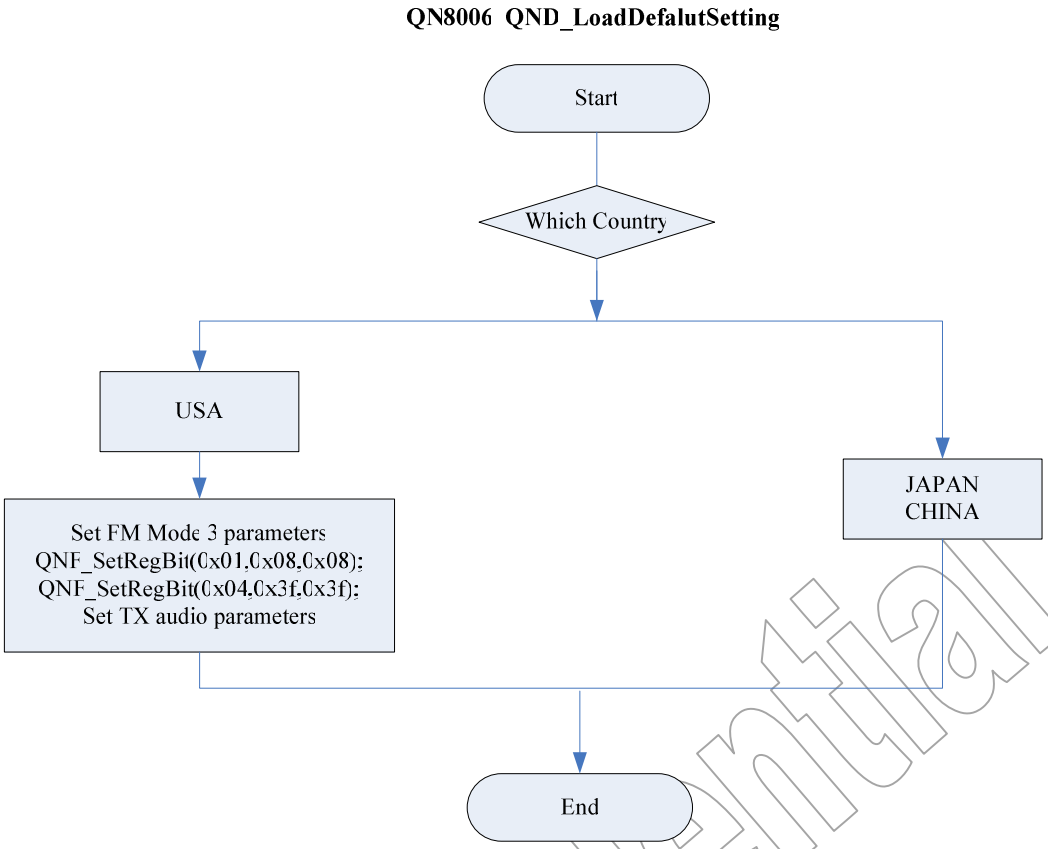
Parameters:

country :COUNTRY_CHINA,COUNTRY_USA,COUNTRY_JAPAN

Return Value:

None

Flowchart:



2.17 QND_TXClearChannelScan

```
**********************************************************************
UINT16          QND_TXClearChannelScan(UINT16 start, UINT16 stop, UINT8 step,UINT8 db)
**********************************************************************
```

Description: Clean channel scan. Finds the best clear channel for transmission.

Parameters:

Start

Set the frequency (10kHz) where scan will be started,
eg: 76.00MHz will be set to 7600.

Stop

Set the frequency (10kHz) where scan will be stopped,
eg: 108.00MHz will be set to 10800.

Step

QND_FSTEP_100KHZ: Set leap step to 100kHz.
QND_FSTEP_200KHZ: Set leap step to 200kHz.
QND_FSTEP_50KHZ: Set leap step to 50kHz.

DB

Set threshold for quality of channel to be searched.

Return Value:

The channel frequency (unit: 10kHz)

Flowchart:

2.18 QND_TXConfigAudio

```
*********************************************************************
void            QND_TXConfigAudio(UINT8 optiontype, UINT8 option )
*********************************************************************
```

Description: Config the Rx Audio performance like volume adjust, mute set.

Parameters:

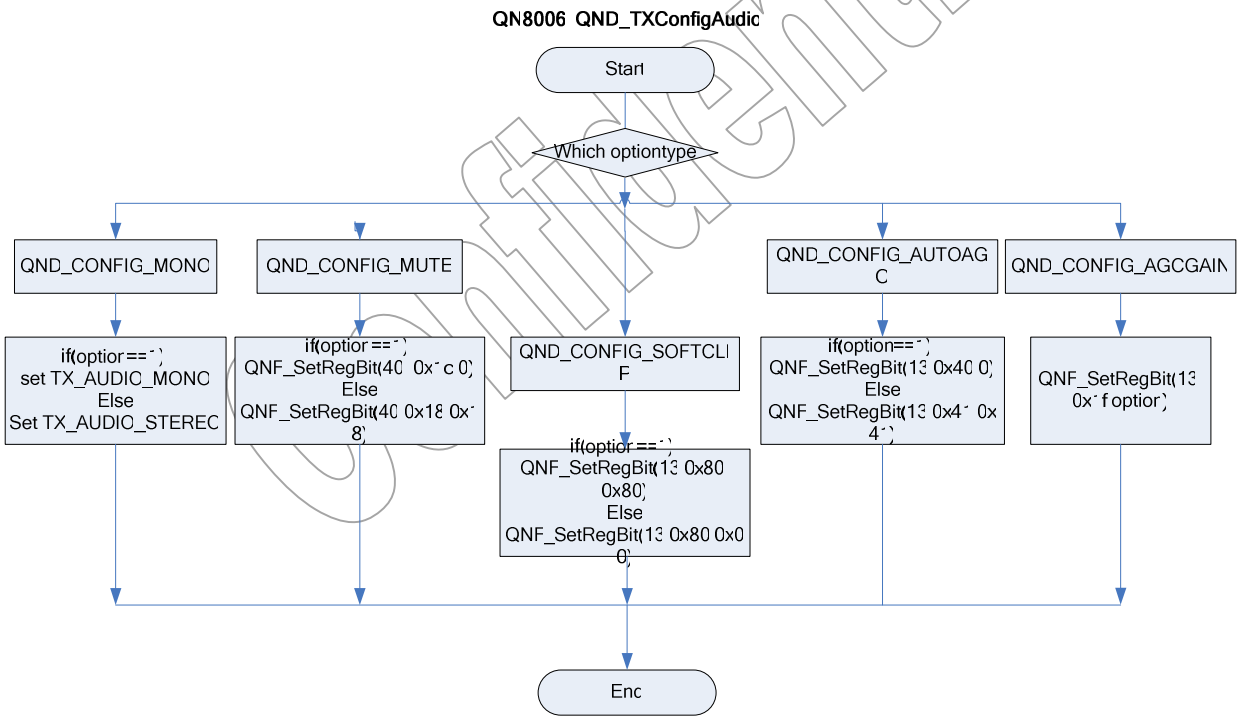
optiontype:

It indicates what you want to config, it can use some marco define to control:
eg: AGCGAINCONFIG, SOFTCLIPCONFIG, MONOCONFIG, OUTPUTFORMATCONFIG
option:
set the configuration value;
if (optiontype== QND_CONFIG_AUTOAGC) : option will control agc,
0:disable agc function, 1:enable function.
if (optiontype== QND_CONFIG_SOFTCLIP); option will control softclip,
0:disable soft clip, 1:enable softclip.
if (optiontype== QND_CONFIG_MONO); option will control mono,
0: QND_AUDIO_STEREO, 1: QND_AUDIO_STEREO
if (optiontype == QND_CONFIG_AGCGAIN); option will control AGC value:0000~1111

Return Value:

None

Flowchart:



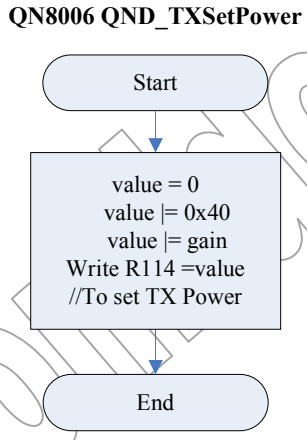
2.19 QND_TXSetPower

```
*********************************************************************
void QND_TXSetPower (UINT8 gain)
*********************************************************************
```

Description: Sets FM transmit power attenuation.
Parameters:
gain
The transmission power attenuation value.
Look Up Table (see below)
00-3F, you can not beyond this range

00: 124	10: 118	20: 112	30: 106
01: 123.25	11: 117.25	21: 111.25	31: 105.25
02: 122.5	12: 116.5	22: 110.5	32: 104.5
03: 121.75	13: 115.75	23: 109.75	33: 103.75
04: 121	14: 115	24: 109	34: 103
05: 120.25	15: 114.25	25: 108.25	35: 102.25
06: 119.5	16: 113.5	26: 107.5	36: 101.5
07: 118.75	17: 112.75	27: 106.75	37: 100.75
08: 118	18: 112	28: 106	38: 100
09: 117.25	19: 111.25	29: 105.25	39: 99.25
0A: 116.5	1A: 110.5	2A: 104.5	3A: 98.5
0B: 115.75	1B: 109.75	2B: 103.75	3B: 97.75
0C: 115	1C: 109	2C: 103	3C: 97
0D: 114.25	1D: 108.25	2D: 102.25	3D: 96.25
0E: 113.5	1E: 107.5	2E: 101.5	3E: 95.5
0F: 112.75	1F: 106.75	2F: 100.75	3F: 94.75

Flowchart:



3 Driver API Code Integration

3.1 Driver API Code

- All driver API code is under project (SDK DEMO GUI) dir: qndriver
- Qndriver.c:
  - Driver API function interface layer
- Qnio.c:
  - Driver API I/O bus layer

3.2 Code Example

```
void QND_SetSysMode(UINT16 mode)
{
    UINT8 val;
    switch(mode)
    {
              case QND_MODE_SLEEP:     //set sleep mode
                QNF_SetRegBit(72, 0x80, 0x00);
                prevMode = QND_READ(SYSTEM1);
                QNF_SetRegBit(SYSTEM1, R_TXRX_MASK, STNBY);
                break;
            case QND_MODE_WAKEUP:   //set wakeup mode
                QND_WriteReg(SYSTEM1, prevMode);
                break;
            case QND_MODE_DEFAULT:
                QNF_SetRegBit(SYSTEM2, 0x30,0x10);
                break;
                …

}
```

3.3 Code Integration

- Specify application CPU catogery (ARM, 8051, …), Quintic chip (8006, 8025, 8066, …) and feature combination (RX/TX/RDS), Quintic FAE will generate qndirver.c/h and qnio.c/h for customer
- Normally only qndriver.c and qndriver.h is required to be integrated into application code if I2C driver code exists
- qndriver.c, qndriver.h
- If I2C driver is not there or selected chip without standard I2C protocol support, then qnio.c and qnio.h is required to be integrated
- qnio.c, qnio.h
- In application code, add:
- #include “qndriver.h"

4 Examples Using Source Code

4.1 Receive a Channel

```
#include "stdio.h"
#include "stdlib.h"
#include "qndriver.h"
void TestMain3()
{
    QND_Init();
    QND_SetSysMode(QND_MODE_FM| QND_MODE_RX);
    QND_TuneToCH(9150);
    printf("Tune to channel 91.50 MHZ\n");
}
```

4.2 Seek Channel

```
#include "stdio.h"
#include "stdlib.h"
#include "qndriver.h"
void TestMain1()
{
    UINT8 nCH;
    UINT8 i;
    QND_Init();
    QND_SetSysMode(QND_MODE_FM| QND_MODE_RX);
    nCH=QND_RXSeekCHAll(7600,10800, QND_FSTEP_100KHZ,6,1);
    if(nCH>0)
```

```
    {
        printf("%d channel(s) found\n",(int)nCH);
        for(i=0;i<nCH;i++)
        {
            printf("CHANNEL %d:%.2f MHZ\n",i,chList[i]/100.0);
        }
    }
    else
    {
        printf("No channel found\n");
    }
}
```

4.3 Seek Next Channel

```
#include "stdio.h"
#include "stdlib.h"
#include "qndriver.h"
void TestMain5()
{
    UINT16 nCH;
    QND_Init();
    QND_SetSysMode(QND_MODE_FM| QND_MODE_RX);
    QND_TuneToCH(9150);
    printf("Current Channel is 91.50 MHZ\n");
    nCH=QND_RXSeekCH(9150+QND_FSTEP_100KHZ,10800,QND_FSTEP_100KHZ,6,1);
    if(nCH!=0)
    {
```

```
            QND_TuneToCH(nCH);
            printf("Next channel is %.2f MHZ\n",nCH/100.0);
        }
        else
        {
            printf("Cannot find valid channel\n");
        }
    }
```

4.4 Show RSSI

```
#include "stdio.h"
#include "stdlib.h"
#include "qndriver.h"
void TestMain2()
{
    UINT16 nStart,nStop,i;
    UINT8 rssi;
    nStart=7600;
    nStop=10800;

    QND_Init();
    QND_SetSysMode(QND_MODE_FM| QND_MODE_RX);
    QND_RXConfigAudio(QND_CONFIG_MUTE,1);
    for(i=nStart;i<=nStop;i++)
    {
        rssi=QND_GetRSSI(i);
        printf("ch=%.2f MHZ,rssi=%d\n",i/100.0,(int)rssi);
```

```
    }
}
```

4.5 Receive RDS Data

```
#include "stdio.h"
#include "stdlib.h"
#include "qndriver.h"
void TestMain8()
{
    UINT8 tmprdsinfo;
    UINT8 register_value;
    UINT8 buffer[8];
    tmprdsinfo=0;
    QND_Init();
    QND_SetSysMode(QND_MODE_FM| QND_MODE_RX);
    QND_TuneToCH(9150);
    QND_RDSEnable(QND_RDS_ON);
    while(1)
    {
            QND_Delay(20);
            register_value = QND_RDSDetectSignal();
            register_value = register_value>>7;
            if(register_value^tmprdsinfo)
            {
                    tmprdsinfo =register_value;
                    QND_RDSLoadData(buffer,0);
```

```
                printf("receive rds data:%02X %02X %02X %02X %02X %02X %02X %02X\n",
                        buffer[0],buffer[1],buffer[2],buffer[3],buffer[4],
                        buffer[5],buffer[6],buffer[7]);
            }
        }
    }
```

4.6 Transmit at a Channel

```
#include "stdio.h"
#include "stdlib.h"
#include "qndriver.h"
void TestMain6()
{
    QND_Init();
    QND_SetSysMode(QND_MODE_FM| QND_MODE_TX);
    QND_TuneToCH(9150);
    QND_TXSetPower(0);
    printf("FM TX at 91.50 MHZ\n");
}
```

4.7 Transmit RDS Data

```
#include "stdio.h"
#include "stdlib.h"
#include "qndriver.h"
void TestMain7()
{
```

```
    UINT8 buffer[8]={0,1,2,3,4,5,6,7};


    QND_Init();
    QND_SetSysMode(QND_MODE_FM| QND_MODE_TX);
    QND_TuneToCH(9150);
    QND_RDSEnable(QND_RDS_ON);
    QND_Delay(15);
    QND_RDSLoadData(buffer,1);
    QND_Delay(20);
    QND_RDSCheckBufferReady();
    printf("Send RAW rds data:00 01 02 03 04 05 06 07 at 91.50MHZ\n");
}
```

Contact Information

Quintic Corporation (USA)
3211 Scott Boulevard, Suite 203
Santa Clara, CA 95054
Tel: +1.408.970.8808
Fax: +1.408.970.8829
Email: support@quinticcorp.com
Web: www.quinticcorp.com

Quintic Microelectronics (China)

Building 8 B-301A Tsinghua Science Park
1st East Zhongguancun Rd, Haidian
Beijing, China 100084
Tel: +86 (10) 8215-1997
Fax: +86 (10) 8215-1570
Email: support@quinticcorp.cn
Web: www.quintic.cn