

CX9261A API

1.7.0

制作者 城芯科技

1 CX9261A驱动

1.1 简介

CX9261A驱动提供CX9261A射频收发器常用API，用户通过调用API实现器件功能的调用与调试。使用时需要包含头文件cx9261adriver.h，并且在cx9261adriver.cpp中提供 writeSpi(), readSpi(), pause() 函数的具体实现

2 命名空间文档

2.1 CX9261A 命名空间参考

类

- struct [AGC](#)
AGC模式的结构体
- struct [AGCDIG](#)
[AGC DIG](#)模式的结构体
- struct [AGCLMT](#)
[AGC LMT](#)模式的结构体
- struct [AGCPDT](#)
[AGC PDT](#)模式的结构体
- struct [Calib](#)
校准相关参数结构体
- struct [CalibPredis](#)
储存窄带校准结果的结构体
- struct [Clock](#)
时钟相关参数结构体
- struct [CX9261AConfiguration](#)
CX9261A结构体
- struct [Filters](#)
配置滤波器相关参数结构体，0代表bypass
- struct [Interface](#)
配置接口相关参数结构体
- struct [PllFreq](#)
配置PLL频率相关参数结构体
- struct [PllSource](#)
配置PLL来源相关参数结构体
- struct [Power](#)
配置电源模式结构体，1代表该模块上电
- struct [PowerManagement](#)
配置电源模式管理，包含模式选择，接收模式，发射模式，FDD模式
- class [Register](#)
[Register](#)类，提供方便的位操作函数
- struct [RxConfig](#)
配置接收通道相关参数结构体
- struct [RxNCO](#)
配置RX NCO相关参数结构体
- struct [TxConfig](#)
配置发射通道相关参数结构体
- struct [TxNCO](#)
配置TX NCO相关参数结构体

枚举

- enum class **WorkMode** : std::uint8_t { **Rx** , **Tx** , **FDD** }
芯片工作模式
- enum class **TrMode** : std::uint8_t { **R1T1** , **R2T2** }
收发数据通道数量
- enum class **PortMode** : std::uint8_t { **Single** , **Dual** }
收发接口通道数量
- enum class **IOMode** : std::uint8_t { **CMOS** , **LVDS_S** , **LVDS_E** }
接口模式: *CMOS CMOS*接口模式, *LVDS_S LVDS*顺序模式, *LVDS_E LVDS*奇偶模式
- enum class **LoSource** : std::uint8_t {
PII1 = 0b00001 , **PII2** = 0b00010 , **PII3** = 0b00100 , **PII4** = 0b01000 ,
PII5 = 0b10000 , **Off** = 0 }
本振来源
- enum class **RxChannel** : std::uint8_t { **Rx1** = 1 , **Rx2** , **Rx3** }
接收通道
- enum class **TxChannel** : std::uint8_t { **Tx1** = 1 , **Tx2** }
发射通道
- enum class **PowerMode** : std::uint8_t {
Shutdown = 0 , **Sleep** , **Alert** , **Calib** ,
Receive , **Transmit** , **Duplex** , **Obs** }
电源模式
- enum class **NCOMode** : std::uint8_t {
Bypass = 0 , **Up** , **Down** , **BIST** ,
Master , **Slaver** }
NCO模式
- enum class **TemperatureComp** : std::uint8_t { **DAC** = 0 , **Band** }
温补模式 *DAC: DAC*模式 *Band: 跳子带*模式
- enum class **CalibMethod** : std::uint8_t { **TST** = 0 , **Bisec** , **MultiPoints** , **Tracking** }
校准方法 *TST: 时域统计 Rx*通道使用, *Bisec: 二分法 Tx*通道使用, *MultiPoints: 多点校准*
- enum class **DCCalibMethod** : std::uint8_t { **Tracking** = 0 , **HighPass** }
*DC*校准方法 *Tracking: 跟踪校准 HighPass: 高通滤波器滤掉直流*

函数

- void **initialChip** ()
初始化芯片状态, 写入与具体配置无关的寄存器值
- void **initialPowerConfig** (const **CX9261AConfiguration** &config)
初始化电源配置, 如果只自定义配置部分模式下的电源管理, 在自定义配置之前需要先调用此函数
- void **initialBBPLL** ()
初始化*BBPLL*相关寄存器, 如果调用了*initialChip*, 则无需单独调用此函数
- void **initialPLL1** ()
初始化*PLL1*相关寄存器, 如果调用了*initialChip*, 则无需单独调用此函数
- void **initialPLL2** ()
初始化*PLL2*相关寄存器, 如果调用了*initialChip*, 则无需单独调用此函数
- void **initialPLL3** ()
初始化*PLL3*相关寄存器, 如果调用了*initialChip*, 则无需单独调用此函数
- void **initialPLL4** ()
初始化*PLL4*相关寄存器, 如果调用了*initialChip*, 则无需单独调用此函数
- void **initialPLL5** ()
初始化*PLL5*相关寄存器, 如果调用了*initialChip*, 则无需单独调用此函数

- void `configRef` (double ref)
配置受参考时钟频率影响的寄存器
- void `configBBPLL` (const `CX9261AConfiguration` &config)
配置时钟相关寄存器，包括ADC和BBPLL
- bool `afcBBPLL` ()
BBPLL AFC
- std::vector< int > `configPIIImpI` (const double lo, const double ref)
- void `configPII1` (double lo, const `Clock` &clock)
配置PLL1频率
- bool `afcPII1` ()
PII1 AFC
- void `configPII2` (double lo, const `Clock` &clock)
配置PLL2频率
- bool `afcPII2` ()
PII2 AFC
- void `configPII3` (double lo, const `Clock` &clock)
配置PLL3频率
- bool `afcPII3` ()
PII3 AFC
- void `configPII4` (double lo, const `Clock` &clock)
配置PLL4频率
- bool `afcPII4` ()
PII4 AFC
- void `configPII5` (double lo, const `Clock` &clock)
配置PLL5频率
- bool `afcPII5` ()
PII5 AFC
- void `configAGCMode` (int mode)
配置AGC模式
- void `configAGCPDT` (const `AGCPDT` &pdt)
配置AGC PDT
- void `configAGCLMT` (const `AGCLMT` &lmt)
配置AGC LMT
- void `configAGCDIG` (const `AGCDIG` &dig)
配置AGC DIG
- void `configAGC` (const `AGC` &agc)
配置AGC
- void `configRISCV` (bool on)
配置片内处理器，仅在使用需要处理器辅助的校准方法时需要开启处理器
- void `configTemperatureCompensation` (`TemperatureComp` mode)
配置温补
- void `configADCCalib` (double adc)
ADC校准
- void `configFlashCalib` ()
Flash校准
- std::map< int, int > `beforeQecState` ()
QEC校准会改变部分寄存器配置，在进行校准之前需要保存这部分寄存器，校准后恢复
- void `restoreQecState` (const std::map< int, int > &state)
恢复QEC校准过程中改变的寄存器
- void `preparePowerForRxCalib` (const `PIISource` &pllSource)
根据pll来源配置校准模式下的电源管理

- void `preparePllForRxCalib` (const `CX9261AConfiguration` &config, `RxChannel` chan)
调整校准时 *Pll* 的配置
- bool `waitRxTSTFinished` ()
等待 *RxTST* 窄带校准结束, 可重写
- void `rxTSTCalib` (`RxChannel` chan)
Rx 时域统计校准
- void `rxDelayCalib` (double fs, `RxChannel` chan)
Rx 延时线校准
- void `configDcCalib` (const `Calib` &config)
DC Tracking
- void `configDcCalibMethod` (const `DCCalibMethod` method)
- void `configDcTrackingLength` (const int length)
- void `configDcHighPassFactor` (const int factor)
- void `rxCalib` (const `CX9261AConfiguration` &config, `RxChannel` chan)
Rx 校准
- void `configCalibMethod` (const `Calib` &calib)
选择校准方法
- `CalibPredis readRxCalibPredis` (`RxChannel` chan)
读取当前 *Rx chan* 通道的校准值
- void `saveRxCalibPredis` (unsigned char index, `RxChannel` chan, `LoSource` pll)
储存校准值
- void `loadRxCalibPredis` (unsigned char index, `RxChannel` chan, `LoSource` pll)
加载保存的校准值
- void `configRxMultiPointsQecImpl` (const `CX9261AConfiguration` &config, const std::vector< double > &points, const `RxChannel` chan)
- void `configRxMultiPointsQec` (const `CX9261AConfiguration` &config, const std::vector< double > &points, `RxChannel` chan)
Rx 窄带多点校准
- void `preparePowerForTxCalib` (const `PllSource` &pllSource)
根据 *pll* 来源配置校准模式下的电源管理
- void `preparePllForTxCalib` (const `CX9261AConfiguration` &config)
调整校准时 *Pll* 的配置
- void `txBisectionQecCalib` (double f0, double fs)
TX 二分法 *QEC* 校准
- void `txTSTCalib` (const `CX9261AConfiguration` &config, `TxChannel` chan)
TX 时域统计法校准
- void `txBisectionDcCalib` (double f0, double fs)
TX 二分法 *DC* 校准
- void `configTxMultiPointsQec` (const `CX9261AConfiguration` &config, const std::vector< double > &points)
Tx 多点校准
- void `txCalib` (const `CX9261AConfiguration` &config)
Tx 通道校准
- void `saveTxCalibPredis` (unsigned char index, `TxChannel` chan)
储存校准值
- void `loadTxCalibPredis` (unsigned char index, `TxChannel` chan, `LoSource` pll)
加载保存的校准值
- void `configQecCalib` (const `CX9261AConfiguration` &config)
Qec 校准顶层函数, 内部根据配置选择使用对应方法进行校准
- void `configTxNco` (const `TxNCO` &nco, double fs)
配置 *TX NCO* 相关寄存器
- void `configTxBIST` (bool bist)

- 配置 *TX NCO BIST* 模式
- void `configTxNcoFreq` (double freq, double fs)
 - 配置 *TX NCO* 频率
- void `configRxNco` (const `RxNCO` &nco, double fs)
 - 配置 *RX NCO* 相关寄存器
- void `configRxNcoMode` (`NCOMode` mode)
 - 配置 *Rx NCO* 模式
- void `configRxNcoMS` (`NCOMode` mode)
 - 配置 *Rx NCO* 模式
- void `configRxNcoFreq` (double freq, double fs)
 - 配置 *Rx NCO* 频率
- void `configInterface` (const `Interface` &interface)
 - 配置接口相关寄存器
- void `configPllSource` (const `PllSource` &pllSource)
 - 配置 *PLL* 来源
- void `configPowerMode` (`PowerMode` pm)
 - 切换电源模式
- void `configPowerPins` (const `Power` &power, `PowerMode` pm)
 - 配置电源管理
- void `configRefDoubler` (bool doubler, double ref)
 - 配置参考倍频器
- void `initialRxChannel` (`RxChannel` chan)
 - 初始化通道, 调用 `initialChip` 函数会初始化 *RX1*, *RX2*, *RX3* 三个通道
- std::pair< int, int > `configBandwidthImpl` (const double bandwidth)
- void `configRxBandwidth` (double bandwidth, `RxChannel` chan)
 - 配置 *Rx* 带宽
- void `configRxTIA` (int tia, `RxChannel` chan)
 - 配置 *Rx TIA*
- void `configRxMixer` (int mixer)
 - 配置 *Rx Mixer*
- void `configRxChannel` (const `RxConfig` &config, `RxChannel` chan)
 - 配置 *Rx* 通道, 包含 *TIA*, *Mixer* 以及 *Bandwidth*
- void `configRxFIR` (int fir)
 - 配置 *Rx FIR* 滤波器抽取倍数
- void `configRxHBF1` (int hbf)
 - 配置 *Rx HBF1* 滤波器抽取倍数
- void `configRxHBF2` (int hbf)
 - 配置 *Rx HBF2* 滤波器抽取倍数
- void `configRxHBF3` (int hbf)
 - 配置 *Rx HBF3* 滤波器抽取倍数
- void `configRxCIC` (int cic)
 - 配置 *Rx CIC* 滤波器
- void `configRxFilters` (const `Filters` &filters)
 - 配置 *Rx* 滤波器
- void `configRxFIRCoe` (const std::vector< double > &coe)
 - 配置 *Rx FIR* 滤波器系数
- std::pair< int, int > `configTxBandwidthImpl` (const double bandwidth)
- void `configTxBandwidth` (double bandwidth, `TxChannel` chan)
 - 配置 *Tx* 带宽
- void `configTxMixer` (int mixer, `TxChannel` chan)
 - 配置 *Tx Mixer*

- void `initialTxChannel` (`TxChannel` chan)
初始化 *Tx* 通道
- void `configTxChannelFrequency` (double lo, `TxChannel` chan)
根据 *Tx* 通道的本振频率，配置对应的参数
- void `configTxChannel` (const `TxConfig` &config, `TxChannel` chan)
配置 *Tx* 通道，*Mixer* 以及 *Bandwidth*
- void `configTxFIR` (int fir)
配置 *Tx FIR* 滤波器插值倍数
- void `configTxHBF1` (int hbf)
配置 *Tx HBF1* 滤波器插值倍数
- void `configTxHBF2` (int hbf)
配置 *Tx HBF2* 滤波器插值倍数
- void `configTxHBF3` (int hbf)
配置 *Tx HBF3* 滤波器插值倍数
- void `configTxCIC` (int cic)
配置 *Tx CIC* 滤波器
- void `configTxFilters` (const `Filters` &filters)
配置 *Tx* 滤波器
- void `configTxFIRCoe` (const std::vector< double > &coe)
配置 *Tx FIR* 滤波器系数
- `CalibPredis readTxCalibPredis` (`TxChannel` chan)
读取当前 *Tx chan* 通道的校准值
- void `configFPGA` (const `Interface` &interface)
此函数为 *Demo* 板配套配置函数，用于配置 *FPGA* 接口，用户开发自己的程序时应重写对应功能
- void `rxFFTCalib` (const double lo, const double fs, const int firLength, const `CX9261AConfiguration` &config, `RxChannel` chan)
- void `txTSTPhaseDelay` (const `Clock` &clock, const double lo, const double fs, const double f0, const `TxChannel` chan)
- void `writeSpi` (int address, int value)
需要实现的函数，实现 *SPI* 写入寄存器功能
- int `readSpi` (int address)
需要实现的函数，实现 *SPI* 读取寄存器功能
- void `pause` (double ms)
需要实现的函数，实现等待功能
- void `setPin` (int pin, int value)
需要实现的函数，实现配置 *FPGA* 对应寄存器功能
- std::string `toString` (const `LoSource` source)
- std::string `toString` (const `RxChannel` chan)
- std::string `toString` (const `TxChannel` chan)
- template<typename T >
constexpr const T & `clamp` (const T &v, const T &lo, const T &hi)
- template<typename Float >
int `round` (Float f)
- template<typename Float >
long long `round64` (Float f)
- template<typename Float >
int `floor` (Float f)
- template<typename Float >
long long `floor64` (Float f)
- template<typename Float >
int `ceil` (Float f)
- template<typename Float >
long long `ceil64` (Float f)
- template<int N>
std::string `frac2bin` (const double frac)

变量

- constexpr double `delay` = 100
AFC等待延时, 单位`ms`
- constexpr double `qecDelay` = 500
QEC等待延时, 单位`ms`

2.1.1 枚举类型说明

CalibMethod

```
enum class CX9261A::CalibMethod : std::uint8_t [strong]
```

校准方法 TST: 时域统计 Rx通道使用, Bisec: 二分法 Tx通道使用, MultiPoints: 多点校准

枚举值

TST	
Bisec	
MultiPoints	
Tracking	

DCCalibMethod

```
enum class CX9261A::DCCalibMethod : std::uint8_t [strong]
```

DC校准方法 Tracking: 跟踪校准 HighPass: 高通滤波器滤掉直流

枚举值

Tracking	
HighPass	

IOMode

```
enum class CX9261A::IOMode : std::uint8_t [strong]
```

接口模式: CMOS CMOS接口模式, LVDS_S LVDS顺序模式, LVDS_E LVDS奇偶模式

枚举值

CMOS	
LVDS↔ _S	
LVDS↔ _E	

LoSource

```
enum class CX9261A::LoSource : std::uint8_t [strong]
```

本振来源

枚举值

PII1	
PII2	
PII3	
PII4	
PII5	
Off	

NCOMode

```
enum class CX9261A::NCOMode : std::uint8_t [strong]
```

NCO模式

枚举值

Bypass	
Up	
Down	
BIST	
Master	
Slaver	

PortMode

```
enum class CX9261A::PortMode : std::uint8_t [strong]
```

收发接口通道数量

枚举值

Single	
Dual	

PowerMode

```
enum class CX9261A::PowerMode : std::uint8_t [strong]
```

电源模式

枚举值

Shutdown	
Sleep	
Alert	
Calib	
Receive	
Transmit	
Duplex	
Obs	

RxChannel

```
enum class CX9261A::RxChannel : std::uint8_t [strong]
```

接收通道

枚举值

Rx1	
Rx2	
Rx3	

TemperatureComp

```
enum class CX9261A::TemperatureComp : std::uint8_t [strong]
```

温补模式 DAC: DAC模式 Band: 跳子带模式

枚举值

DAC	
Band	

TrMode

```
enum class CX9261A::TrMode : std::uint8_t [strong]
```

收发数据通道数量

注解

此模式并不代表使用几个通道进行收发

枚举值

R1T1	
R2T2	

TxChannel

```
enum class CX9261A::TxChannel : std::uint8_t [strong]
```

发射通道

枚举值

Tx1	
Tx2	

WorkMode

```
enum class CX9261A::WorkMode : std::uint8_t [strong]
```

芯片工作模式

枚举值

Rx	
Tx	
FDD	

2.1.2 函数说明

afcBBPll()

```
bool CX9261A::afcBBPll ( )
```

BBPLL AFC

返回

是否锁定

afcPll1()

```
bool CX9261A::afcPll1 ( )
```

Pll1 AFC

返回

是否锁定

afcPII2()

```
bool CX9261A::afcP112 ( )
```

PII2 AFC

返回

是否锁定

afcPII3()

```
bool CX9261A::afcP113 ( )
```

PII3 AFC

返回

是否锁定

afcPII4()

```
bool CX9261A::afcP114 ( )
```

PII4 AFC

返回

是否锁定

afcPII5()

```
bool CX9261A::afcP115 ( )
```

PII5 AFC

返回

是否锁定

beforeQecState()

```
std::map< int, int > CX9261A::beforeQecState ( )
```

QEC校准会改变部分寄存器配置，在进行校准之前需要保存这部分寄存器，校准后恢复

返回

包含寄存器地址和值的map

ceil()

```
template<typename Float >
int CX9261A::ceil (
    Float f )
```

ceil64()

```
template<typename Float >
long long CX9261A::ceil64 (
    Float f )
```

clamp()

```
template<typename T >
constexpr const T & CX9261A::clamp (
    const T & v,
    const T & lo,
    const T & hi ) [constexpr]
```

configADCCalib()

```
void CX9261A::configADCCalib (
    double adc )
```

ADC校准

参数

<i>adc</i>	ADC速率
------------	-------

configAGC()

```
void CX9261A::configAGC (
    const AGC & agc )
```

配置AGC

参数

<i>agc</i>	AGC结构体
------------	--------

configAGCDIG()

```
void CX9261A::configAGCDIG (
    const AGCDIG & dig )
```

配置AGC DIG

参数

<i>dig</i>	AGCDIG结构体
------------	-----------

configAGCLMT()

```
void CX9261A::configAGCLMT (
    const AGCLMT & lmt )
```

配置AGC LMT

参数

<i>lmt</i>	AGCLMT结构体
------------	-----------

configAGCMode()

```
void CX9261A::configAGCMode (
    int mode )
```

配置AGC模式

参数

<i>mode</i>	模式0~13
-------------	--------

configAGCPDT()

```
void CX9261A::configAGCPDT (
    const AGCPDT & pdt )
```

配置AGC PDT

参数

<i>pdt</i>	AGCPDT结构体
------------	-----------

configBandwidthImpl()

```
std::pair< int, int > CX9261A::configBandwidthImpl (
    const double bandwidth )
```

configBBPLL()

```
void CX9261A::configBBPLL (
    const CX9261AConfiguration & config )
```

配置时钟相关寄存器，包括ADC和BBPLL

参数

<i>config</i>	CX9261A参数结构体
---------------	--------------

configCalibMethod()

```
void CX9261A::configCalibMethod (
    const Calib & calib )
```

选择校准方法

参数

<i>calib</i>	校准结构体
--------------	-------

configDcCalib()

```
void CX9261A::configDcCalib (
    const Calib & config )
```

DC Tracking

参数

<i>config</i>	CX9261A配置结构体
---------------	--------------

configDcCalibMethod()

```
void CX9261A::configDcCalibMethod (
    const DCCalibMethod method )
```

configDcHighPassFactor()

```
void CX9261A::configDcHighPassFactor (
    const int factor )
```

configDcTrackingLength()

```
void CX9261A::configDcTrackingLength (
    const int length )
```

configFlashCalib()

```
void CX9261A::configFlashCalib ( )
```

Flash校准

configFPGA()

```
void CX9261A::configFPGA (
    const Interface & interface )
```

此函数为Demo板配套配置函数，用于配置FPGA接口，用户开发自己的程序时应重写对应功能

参数

interface	Interface结构体
-----------	--------------

configInterface()

```
void CX9261A::configInterface (
    const Interface & interface )
```

配置接口相关寄存器

参数

interface	Interface配置结构体
-----------	----------------

configPll1()

```
void CX9261A::configPll1 (
    double lo,
    const Clock & clock )
```

配置PLL1频率

参数

lo	目标频率
clock	时钟结构体，使用其中的ref和refDoubler属性

configPll2()

```
void CX9261A::configPll2 (
    double lo,
    const Clock & clock )
```

配置PLL2频率

参数

<i>lo</i>	目标频率
<i>clock</i>	时钟结构体，使用其中的ref和refDoubler属性

configPll3()

```
void CX9261A::configPll3 (
    double lo,
    const Clock & clock )
```

配置PLL3频率

参数

<i>lo</i>	目标频率
<i>clock</i>	时钟结构体，使用其中的ref和refDoubler属性

configPll4()

```
void CX9261A::configPll4 (
    double lo,
    const Clock & clock )
```

配置PLL4频率

参数

<i>lo</i>	目标频率
<i>clock</i>	时钟结构体，使用其中的ref和refDoubler属性

configPll5()

```
void CX9261A::configPll5 (
    double lo,
    const Clock & clock )
```

配置PLL5频率

参数

<i>lo</i>	目标频率
<i>clock</i>	时钟结构体，使用其中的ref和refDoubler属性

configPllImpl()

```
std::vector< int > CX9261A::configPllImpl (
    const double lo,
    const double ref )
```

configPllSource()

```
void CX9261A::configPllSource (
    const PllSource & pllSource )
```

配置PLL来源

参数

<i>pllSource</i>	PLISource结构体
------------------	--------------

configPowerMode()

```
void CX9261A::configPowerMode (
    PowerMode pm )
```

切换电源模式

参数

<i>pm</i>	电源模式
-----------	------

configPowerPins()

```
void CX9261A::configPowerPins (
    const Power & power,
    PowerMode pm )
```

配置电源管理

参数

<i>power</i>	电源配置结构体
<i>pm</i>	电源模式

configQecCalib()

```
void CX9261A::configQecCalib (
    const CX9261AConfiguration & config )
```

Qec校准顶层函数，内部根据配置选择使用对应方法进行校准

参数

<i>config</i>	CX9261A配置结构体
---------------	--------------

configRef()

```
void CX9261A::configRef (
    double ref )
```

配置受参考时钟频率影响的寄存器

参数

<i>ref</i>	参考时钟频率，单位MHz
------------	--------------

configRefDoubler()

```
void CX9261A::configRefDoubler (
    bool doubler,
    double ref )
```

配置参考倍频器

参数

<i>doubler</i>	倍频器开关
<i>ref</i>	参考时钟频率，单位MHz

configRISCV()

```
void CX9261A::configRISCV (
    bool on )
```

配置片内处理器，仅在使用需要处理器辅助的校准方法时需要开启处理器

参数

<i>on</i>	开启/关闭
-----------	-------

configRxBandwidth()

```
void CX9261A::configRxBandwidth (
    double bandwidth,
    RxChannel chan )
```

配置Rx带宽

注解

只改变Rx带宽会导致通道增益改变，建议使用configRxChannel函数进行带宽调整

参数

<i>bandwidth</i>	带宽，单位(MHz)
<i>chan</i>	通道 Rx1/Rx2/Rx3

configRxChannel()

```
void CX9261A::configRxChannel (
    const RxConfig & config,
    RxChannel chan )
```

配置Rx通道，包含TIA，Mixer以及Bandwidth

参数

<i>config</i>	RxConfig结构体
<i>chan</i>	Rx1/Rx2/Rx3

configRxCIC()

```
void CX9261A::configRxCIC (
    int cic )
```

配置Rx CIC滤波器

参数

<i>cic</i>	抽取倍数 0(Bypass)/20
------------	-------------------

configRxFilters()

```
void CX9261A::configRxFilters (
    const Filters & filters )
```

配置Rx滤波器

参数

<i>filters</i>	Filters结构体
----------------	------------

configRxFIR()

```
void CX9261A::configRxFIR (
    int fir )
```

配置Rx FIR滤波器抽取倍数

参数

<i>fir</i>	抽取倍数 0(Bypass)/1/2/3/4/8
------------	--------------------------

configRxFIRCoe()

```
void CX9261A::configRxFIRCoe (
    const std::vector< double > & coe )
```

配置Rx FIR滤波器系数

参数

<i>coe</i>	滤波器系数，长度为16的整倍数，最大128
------------	-----------------------

configRxHBF1()

```
void CX9261A::configRxHBF1 (
    int hbf )
```

配置Rx HBF1滤波器抽取倍数

参数

<i>hbf</i>	抽取倍数 0(Bypass)/2
------------	------------------

configRxHBF2()

```
void CX9261A::configRxHBF2 (
    int hbf )
```

配置Rx HBF2滤波器抽取倍数

参数

<i>hbf</i>	抽取倍数 0(Bypass)/2
------------	------------------

configRxHBF3()

```
void CX9261A::configRxHBF3 (
    int hbf )
```

配置Rx HBF3滤波器抽取倍数

参数

<i>hbf</i>	抽取倍数 0(Bypass)/2/3
------------	--------------------

configRxMixer()

```
void CX9261A::configRxMixer (
    int mixer )
```

配置Rx Mixer

参数

<i>mixer</i>	mixer档位
--------------	---------

configRxMultiPointsQec()

```
void CX9261A::configRxMultiPointsQec (
    const CX9261AConfiguration & config,
    const std::vector< double > & points,
    RxChannel chan )
```

Rx窄带多点校准

参数

<i>config</i>	CX9261A配置结构体
<i>points</i>	校准点
<i>chan</i>	接收通道

configRxMultiPointsQecImpl()

```
void CX9261A::configRxMultiPointsQecImpl (
    const CX9261AConfiguration & config,
```

```
const std::vector< double > & points,  
const RxChannel chan )
```

configRxNco()

```
void CX9261A::configRxNco (   
    const RxNCO & nco,   
    double fs )
```

配置RX NCO相关寄存器

参数

<i>nco</i>	RX NCO配置结构体
<i>fs</i>	adc速率

configRxNcoFreq()

```
void CX9261A::configRxNcoFreq (   
    double freq,   
    double fs )
```

配置Rx NCO频率

参数

<i>freq</i>	混频频率
<i>fs</i>	adc速率

configRxNcoMode()

```
void CX9261A::configRxNcoMode (   
    NCOMode mode )
```

配置Rx NCO模式

参数

<i>mode</i>	NCO模式: Bypass/BIST/Up/Down
-------------	----------------------------

configRxNcoMS()

```
void CX9261A::configRxNcoMS (   
    NCOMode mode )
```

配置Rx NCO模式

参数

<i>mode</i>	NCO模式: Master/Slaver
-------------	----------------------

configRxTIA()

```
void CX9261A::configRxTIA (
    int tia,
    RxChannel chan )
```

配置Rx TIA

注解

只改变TIA会导致通道带宽改变，建议使用configRxChannel函数进行TIA调整

参数

<i>tia</i>	tia档位
<i>chan</i>	通道 Rx1/Rx2/Rx3

configTemperatureCompensation()

```
void CX9261A::configTemperatureCompensation (
    TemperatureComp mode )
```

配置温补

参数

<i>mode</i>	温补模式 DAC/Band
-------------	---------------

configTxBandwidth()

```
void CX9261A::configTxBandwidth (
    double bandwidth,
    TxChannel chan )
```

配置Tx带宽

参数

<i>bandwidth</i>	带宽，单位(MHz)
<i>chan</i>	通道 TX1/TX2

configTxBandwidthImpl()

```
std::pair< int, int > CX9261A::configTxBandwidthImpl (
    const double bandwidth )
```

configTxBIST()

```
void CX9261A::configTxBIST (
    bool bist )
```

配置TX NCO BIST模式

参数

<i>bist</i>	开启/关闭BIST
-------------	-----------

configTxChannel()

```
void CX9261A::configTxChannel (
    const TxConfig & config,
    TxChannel chan )
```

配置Tx通道，Mixer以及Bandwidth

参数

<i>config</i>	TxConfig结构体
<i>chan</i>	Tx1/Tx2

configTxChannelFrequency()

```
void CX9261A::configTxChannelFrequency (
    double lo,
    TxChannel chan )
```

根据Tx通道的本振频率，配置对应的参数

参数

<i>lo</i>	本振频率，单位(MHz)
<i>chan</i>	通道(Tx1/Tx2)

configTxCIC()

```
void CX9261A::configTxCIC (
    int cic )
```

配置Tx CIC滤波器

参数

<i>cic</i>	插值倍数 0(Bypass)/8/20/32/80
------------	---------------------------

configTxFilters()

```
void CX9261A::configTxFilters (
    const Filters & filters )
```

配置Tx滤波器

参数

<i>filters</i>	Filters结构体
----------------	------------

configTxFIR()

```
void CX9261A::configTxFIR (
    int fir )
```

配置Tx FIR滤波器插值倍数

参数

<i>fir</i>	插值倍数 0(Bypass)/1/2/3/4/8
------------	--------------------------

configTxFIRCoe()

```
void CX9261A::configTxFIRCoe (
    const std::vector< double > & coe )
```

配置Tx FIR滤波器系数

参数

<i>coe</i>	滤波器系数，长度为16的整倍数，最大128
------------	-----------------------

configTxHBF1()

```
void CX9261A::configTxHBF1 (
    int hbf )
```

配置Tx HBF1滤波器插值倍数

参数

<i>hbf</i>	插值倍数 0(Bypass)/2
------------	------------------

configTxHBF2()

```
void CX9261A::configTxHBF2 (
    int hbf )
```

配置Tx HBF2滤波器插值倍数

参数

<i>hbf</i>	插值倍数 0(Bypass)/2
------------	------------------

configTxHBF3()

```
void CX9261A::configTxHBF3 (
    int hbf )
```

配置Tx HBF3滤波器插值倍数

参数

<i>hbf</i>	插值倍数 0(Bypass)/2/3
------------	--------------------

configTxMixer()

```
void CX9261A::configTxMixer (
    int mixer,
    TxChannel chan )
```

配置Tx Mixer

参数

<i>mixer</i>	混频器值
<i>chan</i>	TX1/TX2

configTxMultiPointsQec()

```
void CX9261A::configTxMultiPointsQec (
    const CX9261AConfiguration & config,
    const std::vector< double > & points )
```

Tx多点校准

参数

<i>config</i>	CX9261A配置结构体
<i>points</i>	校准点

configTxNco()

```
void CX9261A::configTxNco (
    const TxNCO & nco,
    double fs )
```

配置TX NCO相关寄存器

参数

<i>nco</i>	TX NCO配置结构体
<i>fs</i>	adc速率

configTxNcoFreq()

```
void CX9261A::configTxNcoFreq (
    double freq,
    double fs )
```

配置TX NCO频率

参数

<i>freq</i>	混频频率
<i>fs</i>	adc抽取后速率

floor()

```
template<typename Float >
int CX9261A::floor (
    Float f )
```

floor64()

```
template<typename Float >
long long CX9261A::floor64 (
    Float f )
```

frac2bin()

```
template<int N>
std::string CX9261A::frac2bin (
    const double frac )
```

initialBBPLL()

```
void CX9261A::initialBBPLL ( )
```

初始化BBPLL相关寄存器，如果调用了initialChip，则无需单独调用此函数

initialChip()

```
void CX9261A::initialChip ( )
```

初始化芯片状态，写入与具体配置无关的寄存器值

initialPLL1()

```
void CX9261A::initialPLL1 ( )
```

初始化PLL1相关寄存器，如果调用了initialChip，则无需单独调用此函数

initialPLL2()

```
void CX9261A::initialPLL2 ( )
```

初始化PLL2相关寄存器，如果调用了initialChip，则无需单独调用此函数

initialPLL3()

```
void CX9261A::initialPLL3 ( )
```

初始化PLL3相关寄存器，如果调用了initialChip，则无需单独调用此函数

initialPLL4()

```
void CX9261A::initialPLL4 ( )
```

初始化PLL4相关寄存器，如果调用了initialChip，则无需单独调用此函数

initialPLL5()

```
void CX9261A::initialPLL5 ( )
```

初始化PLL5相关寄存器，如果调用了initialChip，则无需单独调用此函数

initialPowerConfig()

```
void CX9261A::initialPowerConfig (
    const CX9261AConfiguration & config )
```

初始化电源配置，如果只自定义配置部分模式下的电源管理，在自定义配置之前需要先调用此函数

initialRxChannel()

```
void CX9261A::initialRxChannel (
    RxChannel chan )
```

初始化通道，调用initialChip函数会初始化RX1，RX2，RX3三个通道

参数

chan	通道 RX1/RX2/RX3
------	----------------

initialTxChannel()

```
void CX9261A::initialTxChannel (
    TxChannel chan )
```

初始化Tx通道

参数

chan	通道 TX1/TX2
------	------------

loadRxCalibPredis()

```
void CX9261A::loadRxCalibPredis (
    unsigned char index,
    RxChannel chan,
    LoSource pll )
```

加载保存的校准值

参数

index	第index组校准值
chan	指定加载校准值的通道
pll	使用的PLL

loadTxCalibPredis()

```
void CX9261A::loadTxCalibPredis (
    unsigned char index,
    TxChannel chan,
    LoSource pll )
```

加载保存的校准值

参数

<i>index</i>	第index组校准值
<i>chan</i>	指定加载校准值的通道
<i>pll</i>	指定加载校准值对应的PLL

pause()

```
void CX9261A::pause (
    double ms )
```

需要实现的函数，实现等待功能

参数

<i>ms</i>	等待时间，单位毫秒
-----------	-----------

preparePllForRxCalib()

```
void CX9261A::preparePllForRxCalib (
    const CX9261AConfiguration & config,
    RxChannel chan )
```

调整校准时Pll的配置

参数

<i>config</i>	CX9261A配置结构体
<i>chan</i>	要进行校准的通道

preparePllForTxCalib()

```
void CX9261A::preparePllForTxCalib (
    const CX9261AConfiguration & config )
```

调整校准时Pll的配置

参数

<i>config</i>	CX9261A配置结构体
---------------	--------------

preparePowerForRxCalib()

```
void CX9261A::preparePowerForRxCalib (
    const PllSource & pllSource )
```

根据pll来源配置校准模式下的电源管理

参数

<i>pllSource</i>	PllSource结构体
------------------	--------------

preparePowerForTxCalib()

```
void CX9261A::preparePowerForTxCalib (
    const PllSource & pllSource )
```

根据pll来源配置校准模式下的电源管理

参数

<i>pllSource</i>	PllSource结构体
------------------	--------------

readRxCalibPredis()

```
CalibPredis CX9261A::readRxCalibPredis (
    RxChannel chan )
```

读取当前Rx chan通道的校准值

参数

<i>chan</i>	Rx通道
-------------	------

返回

校准值结构体

readSpi()

```
int CX9261A::readSpi (
    int address )
```

需要实现的函数，实现SPI读取寄存器功能

参数

<i>address</i>	要读取的寄存器地址
----------------	-----------

返回

返回的寄存器值

readTxCalibPredis()

```
CalibPredis CX9261A::readTxCalibPredis (
    TxChannel chan )
```

读取当前Tx chan通道的校准值

参数

<i>chan</i>	Tx通道
-------------	------

返回

校准值结构体

restoreQecState()

```
void CX9261A::restoreQecState (
    const std::map< int, int > & state )
```

恢复QEC校准过程中改变的寄存器

参数

<i>state</i>	beforeQecState函数返回的map
--------------	------------------------

round()

```
template<typename Float >
int CX9261A::round (
    Float f )
```

round64()

```
template<typename Float >
long long CX9261A::round64 (
    Float f )
```

rxCalib()

```
void CX9261A::rxCalib (
    const CX9261AConfiguration & config,
    RxChannel chan )
```

Rx校准

参数

<i>config</i>	CX9261A配置结构体
<i>chan</i>	要进行校准的通道

rxDelayCalib()

```
void CX9261A::rxDelayCalib (
    double fs,
    RxChannel chan )
```

RX延时线校准

参数

<i>fs</i>	ADC采样率
<i>chan</i>	校准的RX通道

rxFFTCalib()

```
void CX9261A::rxFFTCalib (
    const double lo,
    const double fs,
    const int firLength,
    const CX9261AConfiguration & config,
    RxChannel chan )
```

rxTSTCalib()

```
void CX9261A::rxTSTCalib (
    RxChannel chan )
```

RX时域统计校准

参数

<i>chan</i>	校准的RX通道
-------------	---------

saveRxCalibPredis()

```
void CX9261A::saveRxCalibPredis (
    unsigned char index,
    RxChannel chan,
    LoSource pll )
```

储存校准值

参数

<i>index</i>	第index组校准值
<i>chan</i>	接收通道
<i>pll</i>	使用的PLL

saveTxCalibPredis()

```
void CX9261A::saveTxCalibPredis (
    unsigned char index,
    TxChannel chan )
```

储存校准值

参数

<i>index</i>	第index组校准值
<i>chan</i>	发射通道

setPin()

```
void CX9261A::setPin (
    int pin,
    int value )
```

需要实现的函数，实现配置FPGA对应寄存器功能

参数

<i>pin</i>	寄存器地址
<i>value</i>	寄存器值

toString() [1/3]

```
std::string CX9261A::toString (
    const LoSource source ) [inline]
```

toString() [2/3]

```
std::string CX9261A::toString (
    const RxChannel chan ) [inline]
```

toString() [3/3]

```
std::string CX9261A::toString (
    const TxChannel chan ) [inline]
```

txBisectionDcCalib()

```
void CX9261A::txBisectionDcCalib (
    double f0,
    double fs )
```

TX二分法DC校准

参数

<i>f0</i>	校准音频率
<i>fs</i>	adc频率

txBisectionQecCalib()

```
void CX9261A::txBisectionQecCalib (
    double f0,
    double fs )
```

TX二分法QEC校准

参数

<i>f0</i>	校准音频率
<i>fs</i>	adc频率

txCalib()

```
void CX9261A::txCalib (
    const CX9261AConfiguration & config )
```

Tx通道校准

参数

<i>config</i>	CX9261A配置结构体
---------------	--------------

txTSTCalib()

```
void CX9261A::txTSTCalib (
    const CX9261AConfiguration & config,
    TxChannel chan )
```

TX时域统计法校准

参数

<i>config</i>	CX9261A配置结构体
<i>chan</i>	要校准的通道

txTSTPhaseDelay()

```
void CX9261A::txTSTPhaseDelay (
    const Clock & clock,
    const double lo,
    const double fs,
    const double f0,
    const TxChannel chan )
```

waitRxTSTFinished()

```
bool CX9261A::waitRxTSTFinished ( )
```

等待RxTST窄带校准结束，可重写

返回

校准是否结束

writeSpi()

```
void CX9261A::writeSpi (
    int address,
    int value )
```

需要实现的函数，实现SPI写入寄存器功能

参数

<i>address</i>	要写入的寄存器地址
<i>value</i>	要写入的寄存器值

2.1.3 变量说明

delay

```
constexpr double CX9261A::delay = 100 [constexpr]
```

AFC等待延时，单位ms

qecDelay

```
constexpr double CX9261A::qecDelay = 500 [constexpr]
```

QEC等待延时，单位ms

3 类说明

3.1 CX9261A::AGC结构体 参考

AGC模式的结构体

```
#include <defs.h>
```

Public 属性

- int [mode](#)
AGC模式选择
- [AGCPDT pdt](#)
AGC PDT模式的结构体
- [AGCLMT lmt](#)
AGC LMT模式的结构体
- [AGCDIG dig](#)
AGC DIG模式的结构体

3.1.1 详细描述

AGC模式的结构体

3.1.2 类成员变量说明

dig

```
AGCDIG CX9261A::AGC::dig
```

[AGC DIG](#)模式的结构体

lmt

`AGCLMT CX9261A::AGC::lmt`

AGC LMT模式的结构体

mode

`int CX9261A::AGC::mode`

AGC模式选择

pdt

`AGCPDT CX9261A::AGC::pdt`

AGC PDT模式的结构体

3.2 CX9261A::AGCDIG结构体 参考

AGC DIG模式的结构体

```
#include <defs.h>
```

Public 属性

- `int count`
DIG统计次数阈值
- `int threshold1`
DIG阈值第一档
- `int threshold2`
DIG阈值第二档
- `int threshold3`
DIG阈值第三档
- `int threshold4`
DIG阈值第四档

3.2.1 详细描述

AGC DIG模式的结构体

3.2.2 类成员变量说明**count**

`int CX9261A::AGCDIG::count`

DIG统计次数阈值

threshold1

```
int CX9261A::AGCDIG::threshold1
```

DIG阈值第一档

threshold2

```
int CX9261A::AGCDIG::threshold2
```

DIG阈值第二档

threshold3

```
int CX9261A::AGCDIG::threshold3
```

DIG阈值第三档

threshold4

```
int CX9261A::AGCDIG::threshold4
```

DIG阈值第四档

3.3 CX9261A::AGCLMT结构体 参考

AGC LMT模式的结构体

```
#include <defs.h>
```

Public 属性

- int [count](#)
*LMT*统计次数阈值
- int [threshold1](#)
*LMT*阈值第一档
- int [threshold2](#)
*LMT*阈值第二档
- int [threshold3](#)
*LMT*阈值第三档
- int [threshold4](#)
*LMT*阈值第四档

3.3.1 详细描述

AGC LMT模式的结构体

3.3.2 类成员变量说明

count

```
int CX9261A::AGCLMT::count
```

LMT统计次数阈值

threshold1

```
int CX9261A::AGCLMT::threshold1
```

LMT阈值第一档

threshold2

```
int CX9261A::AGCLMT::threshold2
```

LMT阈值第二档

threshold3

```
int CX9261A::AGCLMT::threshold3
```

LMT阈值第三档

threshold4

```
int CX9261A::AGCLMT::threshold4
```

LMT阈值第四档

3.4 CX9261A::AGCPDT结构体 参考

AGC PDT模式的结构体

```
#include <defs.h>
```

Public 属性

- int [pdtN](#)
- int [pdtM](#)

3.4.1 详细描述

AGC PDT模式的结构体

3.4.2 类成员变量说明

pdtM

```
int CX9261A::AGCPDT::pdtM
```

pdtN

```
int CX9261A::AGCPDT::pdtN
```

3.5 CX9261A::Calib结构体 参考

校准相关参数结构体

```
#include <defs.h>
```

Public 属性

- bool [calibDC](#)
是否开启 *DC Tracking*
- [DCCalibMethod](#) [dcCalibMethod](#)
- int [dcTrackingLength](#)
*DC Tracking*长度
- int [dcHighPassFactor](#)
*DC*高通滤波系数，越大代表滤波器阶数越高，滤波范围越窄
- std::uint8_t [rx1CalibPll](#)
*rx1*校准的 *pll*选择，*Pll1* | *Pll2*
- std::uint8_t [rx2CalibPll](#)
*rx2*校准的 *pll*选择，*Pll1* | *Pll2*
- std::uint8_t [rx3CalibPll](#)
*rx3*校准的 *pll*选择，*Pll5*
- std::uint8_t [tx1CalibPll](#)
*tx1*校准的 *pll*选择，*Pll1* | *Pll2* | *Pll3* | *Pll4*
- std::uint8_t [tx2CalibPll](#)
*tx2*校准的 *pll*选择，*Pll1* | *Pll2* | *Pll3* | *Pll4*
- [CalibMethod](#) [rxMethod](#)
- [CalibMethod](#) [txMethod](#)
- double [f0](#)
校准音频率
- std::vector< double > [rxMultiPoints](#)
*Rx*多点校准的频率
- std::vector< double > [txMultiPoints](#)
*Tx*多点校准的频率

3.5.1 详细描述

校准相关参数结构体

3.5.2 类成员变量说明

calibDC

```
bool CX9261A::Calib::calibDC
```

是否开启DC Tracking

dcCalibMethod

```
DCCalibMethod CX9261A::Calib::dcCalibMethod
```

dcHighPassFactor

```
int CX9261A::Calib::dcHighPassFactor
```

DC高通滤波系数，越大代表滤波器阶数越高，滤波范围越窄

dcTrackingLength

```
int CX9261A::Calib::dcTrackingLength
```

DC Tracking长度

f0

```
double CX9261A::Calib::f0
```

校准音频率

rx1CalibPll

```
std::uint8_t CX9261A::Calib::rx1CalibPll
```

rx1校准的pll选择，Pll1 | Pll2

rx2CalibPll

```
std::uint8_t CX9261A::Calib::rx2CalibPll
```

rx2校准的pll选择，Pll1 | Pll2

rx3CalibPll

```
std::uint8_t CX9261A::Calib::rx3CalibPll
```

rx3校准的pll选择，Pll5

rxMethod

```
CalibMethod CX9261A::Calib::rxMethod
```

rxMultiPoints

```
std::vector<double> CX9261A::Calib::rxMultiPoints
```

Rx多点校准的频率

tx1CalibPll

```
std::uint8_t CX9261A::Calib::tx1CalibPll
```

tx1校准的pll选择, Pll1 | Pll2 | Pll3 | Pll4

tx2CalibPll

```
std::uint8_t CX9261A::Calib::tx2CalibPll
```

tx2校准的pll选择, Pll1 | Pll2 | Pll3 | Pll4

txMethod

```
CalibMethod CX9261A::Calib::txMethod
```

txMultiPoints

```
std::vector<double> CX9261A::Calib::txMultiPoints
```

Tx多点校准的频率

3.6 CX9261A::CalibPredis结构体 参考

储存窄带校准结果的结构体

```
#include <defs.h>
```

Public 属性

- int [bHigh](#)
- int [bLow](#)
- int [cHigh](#)
- int [cLow](#)
- int [dciHigh](#)
- int [dciLow](#)
- int [dcqHigh](#)
- int [dcqLow](#)

3.6.1 详细描述

储存窄带校准结果的结构体

3.6.2 类成员变量说明

bHigh

```
int CX9261A::CalibPredis::bHigh
```

bLow

```
int CX9261A::CalibPredis::bLow
```

cHigh

```
int CX9261A::CalibPredis::cHigh
```

cLow

```
int CX9261A::CalibPredis::cLow
```

dciHigh

```
int CX9261A::CalibPredis::dciHigh
```

dciLow

```
int CX9261A::CalibPredis::dciLow
```

dcqHigh

```
int CX9261A::CalibPredis::dcqHigh
```

dcqLow

```
int CX9261A::CalibPredis::dcqLow
```

3.7 CX9261A::Clock结构体 参考

时钟相关参数结构体

```
#include <defs.h>
```

Public 属性

- double [ref](#)
参考时钟频率，单位MHz
- double [adc](#)
ADC频率，单位MHz
- int [refDiv](#)
参考时钟分频比
- bool [bbUseIntDiv](#)
是否使用整数分频模式
- bool [refDoubler](#)
是否使用参考时钟倍频器

3.7.1 详细描述

时钟相关参数结构体

3.7.2 类成员变量说明

adc

```
double CX9261A::Clock::adc
```

ADC频率，单位MHz

bbUseIntDiv

```
bool CX9261A::Clock::bbUseIntDiv
```

是否使用整数分频模式

ref

```
double CX9261A::Clock::ref
```

参考时钟频率，单位MHz

refDiv

```
int CX9261A::Clock::refDiv
```

参考时钟分频比

refDoubler

```
bool CX9261A::Clock::refDoubler
```

是否使用参考时钟倍频器

3.8 CX9261A::CX9261AConfiguration结构体 参考

CX9261A结构体

```
#include <defs.h>
```

Public 属性

- [TemperatureComp tempComp](#)
温度补偿
- [AGC agc](#)
AGC结构体
- [Interface interface](#)
Interface结构体
- [Clock clock](#)
Clock结构体
- [PIISource pllSource](#)
PIISource结构体
- [Calib calib](#)
Calib结构体
- [PIIFreq pllFreq](#)
PIIFreq结构体
- [TxNCO txNco](#)
TxNCO结构体
- [RxNCO rxNco](#)
RxNCO结构体
- [RxConfig rx1Config](#)
RxConfig结构体
- [RxConfig rx2Config](#)
RxConfig结构体
- [RxConfig rx3Config](#)
RxConfig结构体
- [TxConfig tx1Config](#)
TxConfig结构体
- [TxConfig tx2Config](#)
TxConfig结构体
- [Filters rxFilters](#)
Filters结构体
- [Filters txFilters](#)
Filters结构体
- [PowerManagement powerManagement](#)
PowerManagement结构体
- `std::vector< double > rxFIRCoe`
Rx FIR滤波器系数
- `std::vector< double > txFIRCoe`
Tx FIR滤波器系数

3.8.1 详细描述

CX9261A结构体

3.8.2 类成员变量说明

agc

`AGC CX9261A::CX9261AConfiguration::agc`

AGC结构体

calib

`Calib CX9261A::CX9261AConfiguration::calib`

Calib结构体

clock

`Clock CX9261A::CX9261AConfiguration::clock`

Clock结构体

interface

`Interface CX9261A::CX9261AConfiguration::interface`

Interface结构体

pllFreq

`PllFreq CX9261A::CX9261AConfiguration::pllFreq`

PllFreq结构体

pllSource

`PllSource CX9261A::CX9261AConfiguration::pllSource`

PllSource结构体

powerManagement

`PowerManagement CX9261A::CX9261AConfiguration::powerManagement`

PowerManagement结构体

rx1Config

`RxConfig` CX9261A::CX9261AConfiguration::rx1Config

RxConfig结构体

rx2Config

`RxConfig` CX9261A::CX9261AConfiguration::rx2Config

RxConfig结构体

rx3Config

`RxConfig` CX9261A::CX9261AConfiguration::rx3Config

RxConfig结构体

rxFilters

`Filters` CX9261A::CX9261AConfiguration::rxFilters

Filters结构体

rxFIRCoe

`std::vector<double>` CX9261A::CX9261AConfiguration::rxFIRCoe

Rx FIR滤波器系数

rxNco

`RxNCO` CX9261A::CX9261AConfiguration::rxNco

RxNCO结构体

tempComp

`TemperatureComp` CX9261A::CX9261AConfiguration::tempComp

温度补偿

tx1Config

`TxConfig` CX9261A::CX9261AConfiguration::tx1Config

TxConfig结构体

tx2Config

`TxConfig` CX9261A::CX9261AConfiguration::tx2Config

TxConfig结构体

txFilters

`Filters` CX9261A::CX9261AConfiguration::txFilters

Filters结构体

txFIRCoe

`std::vector<double>` CX9261A::CX9261AConfiguration::txFIRCoe

Tx FIR滤波器系数

txNco

`TxNCO` CX9261A::CX9261AConfiguration::txNco

TxNCO结构体

3.9 CX9261A::Filters结构体 参考

配置滤波器相关参数结构体，0代表bypass

```
#include <defs.h>
```

Public 属性

- int `fir`
*FIR*滤波器抽取/插值倍数
- int `hbf1`
*HBF1*滤波器抽取/插值倍数
- int `hbf2`
*HBF2*滤波器抽取/插值倍数
- int `hbf3`
*HBF3*滤波器抽取/插值倍数
- int `cic`
*CIC*滤波器抽取/插值倍数

3.9.1 详细描述

配置滤波器相关参数结构体，0代表bypass

3.9.2 类成员变量说明

cic

```
int CX9261A::Filters::cic
```

CIC滤波器抽取/插值倍数

fir

```
int CX9261A::Filters::fir
```

FIR滤波器抽取/插值倍数

hbf1

```
int CX9261A::Filters::hbf1
```

HBF1滤波器抽取/插值倍数

hbf2

```
int CX9261A::Filters::hbf2
```

HBF2滤波器抽取/插值倍数

hbf3

```
int CX9261A::Filters::hbf3
```

HBF3滤波器抽取/插值倍数

3.10 CX9261A::Interface结构体 参考

配置接口相关参数结构体

```
#include <defs.h>
```

Public 属性

- [WorkMode workMode](#)
芯片工作模式
- [TrMode trMode](#)
收发数据通道模式
- [PortMode portMode](#)
收发接口通道模式
- [IOMode ioMode](#)
接口电平模式

3.10.1 详细描述

配置接口相关参数结构体

3.10.2 类成员变量说明

ioMode

`IOMode` CX9261A::Interface::ioMode

接口电平模式

portMode

`PortMode` CX9261A::Interface::portMode

收发接口通道模式

trMode

`TrMode` CX9261A::Interface::trMode

收发数据通道模式

workMode

`WorkMode` CX9261A::Interface::workMode

芯片工作模式

3.11 CX9261A::PIIFreq结构体 参考

配置PLL频率相关参数结构体

```
#include <defs.h>
```

Public 属性

- double `pll1Freq`
PLL1本振时钟，单位MHz
- double `pll2Freq`
PLL2本振时钟，单位MHz
- double `pll3Freq`
PLL3本振时钟，单位MHz
- double `pll4Freq`
PLL4本振时钟，单位MHz
- double `pll5Freq`
PLL5本振时钟，单位MHz
- double `rxExtFreq`
接收通道使用外本振时，外本振频率，单位MHz
- double `txExtFreq`
发射通道使用外本振时，外本振频率，单位MHz

3.11.1 详细描述

配置PLL频率相关参数结构体

3.11.2 类成员变量说明

pll1Freq

```
double CX9261A::PllFreq::pll1Freq
```

PLL1本振时钟，单位MHz

pll2Freq

```
double CX9261A::PllFreq::pll2Freq
```

PLL2本振时钟，单位MHz

pll3Freq

```
double CX9261A::PllFreq::pll3Freq
```

PLL3本振时钟，单位MHz

pll4Freq

```
double CX9261A::PllFreq::pll4Freq
```

PLL4本振时钟，单位MHz

pll5Freq

```
double CX9261A::PllFreq::pll5Freq
```

PLL5本振时钟，单位MHz

rxExtFreq

```
double CX9261A::PllFreq::rxExtFreq
```

接收通道使用外本振时，外本振频率，单位MHz

txExtFreq

```
double CX9261A::PllFreq::txExtFreq
```

发射通道使用外本振时，外本振频率，单位MHz

3.12 CX9261A::PllSource结构体 参考

配置PLL来源相关参数结构体

```
#include <defs.h>
```

Public 属性

- bool [rxUseExt](#)
接收通道是否使用外本振
- bool [txUseExt](#)
发射通道是否使用外本振
- [LoSource rx1Source](#)
*Rx1*通道本振来源
- [LoSource rx2Source](#)
*Rx2*通道本振来源
- [LoSource rx3Source](#)
*Rx3*通道本振来源
- [LoSource tx1Source](#)
*Tx1*通道本振来源
- [LoSource tx2Source](#)
*Tx2*通道本振来源

3.12.1 详细描述

配置PLL来源相关参数结构体

3.12.2 类成员变量说明

rx1Source

```
LoSource CX9261A::PllSource::rx1Source
```

Rx1通道本振来源

rx2Source

```
LoSource CX9261A::PllSource::rx2Source
```

Rx2通道本振来源

rx3Source

```
LoSource CX9261A::PllSource::rx3Source
```

Rx3通道本振来源

rxUseExt

```
bool CX9261A::PllSource::rxUseExt
```

接收通道是否使用外本振

tx1Source

```
LoSource CX9261A::PllSource::tx1Source
```

Tx1通道本振来源

tx2Source

```
LoSource CX9261A::PllSource::tx2Source
```

Tx2通道本振来源

txUseExt

```
bool CX9261A::PllSource::txUseExt
```

发射通道是否使用外本振

3.13 CX9261A::Power结构体 参考

配置电源模式结构体，1代表该模块上电

```
#include <defs.h>
```

Public 属性

- bool [dig_tx2](#)
- bool [dig_tx1](#)
- bool [dig_rx3](#)
- bool [dig_rx2](#)
- bool [dig_rx1](#)
- bool [fref1_rx](#)
- bool [auxadc2](#)
- bool [auxadc1](#)
- bool [auxdac](#)
- bool [temp_sensor](#)
- bool [bgr](#)
- bool [exlo](#)
- bool [tx2](#)
- bool [tx1](#)
- bool [rx3](#)
- bool [rx2](#)

- bool rx1
- bool pll5_lldo
- bool pll5
- bool pll4_lldo
- bool pll4
- bool pll3_lldo
- bool pll3
- bool pll2_lldo
- bool pll2
- bool pll1_lldo
- bool pll1
- bool bbpll

3.13.1 详细描述

配置电源模式结构体，1代表该模块上电

3.13.2 类成员变量说明

auxadc1

```
bool CX9261A::Power::auxadc1
```

auxadc2

```
bool CX9261A::Power::auxadc2
```

auxdac

```
bool CX9261A::Power::auxdac
```

bbpll

```
bool CX9261A::Power::bbpll
```

bgr

```
bool CX9261A::Power::bgr
```

dig_rx1

```
bool CX9261A::Power::dig_rx1
```

dig_rx2

```
bool CX9261A::Power::dig_rx2
```

dig_rx3

```
bool CX9261A::Power::dig_rx3
```

dig_tx1

```
bool CX9261A::Power::dig_tx1
```

dig_tx2

```
bool CX9261A::Power::dig_tx2
```

exlo

```
bool CX9261A::Power::exlo
```

fref1_rx

```
bool CX9261A::Power::fref1_rx
```

pll1

```
bool CX9261A::Power::pll1
```

pll1_ldo

```
bool CX9261A::Power::pll1_ldo
```

pll2

```
bool CX9261A::Power::pll2
```

pll2_ldo

```
bool CX9261A::Power::pll2_ldo
```

pll3

```
bool CX9261A::Power::pll3
```

pll3_ldo

```
bool CX9261A::Power::pll3_ldo
```

pll4

```
bool CX9261A::Power::pll4
```

pll4_ldo

```
bool CX9261A::Power::pll4_ldo
```

pll5

```
bool CX9261A::Power::pll5
```

pll5_ldo

```
bool CX9261A::Power::pll5_ldo
```

rx1

```
bool CX9261A::Power::rx1
```

rx2

```
bool CX9261A::Power::rx2
```

rx3

```
bool CX9261A::Power::rx3
```

temp_sensor

```
bool CX9261A::Power::temp_sensor
```

tx1

```
bool CX9261A::Power::tx1
```

tx2

```
bool CX9261A::Power::tx2
```

3.14 CX9261A::PowerManagement结构体 参考

配置电源模式管理，包含模式选择，接收模式，发射模式，FDD模式

```
#include <defs.h>
```

Public 属性

- [PowerMode powerMode](#)
- [Power rxPower](#)
- [Power txPower](#)
- [Power fddPower](#)

3.14.1 详细描述

配置电源模式管理，包含模式选择，接收模式，发射模式，FDD模式

3.14.2 类成员变量说明**fddPower**

```
Power CX9261A::PowerManagement::fddPower
```

powerMode

```
PowerMode CX9261A::PowerManagement::powerMode
```

rxPower

```
Power CX9261A::PowerManagement::rxPower
```

txPower

```
Power CX9261A::PowerManagement::txPower
```

3.15 CX9261A::Register类 参考

Register类，提供方便的位操作函数

```
#include <utils.h>
```

Public 成员函数

- `template<typename T = int>`
`constexpr Register (const T value=0)`
- `unsigned char value () const`
以十进制返回寄存器值
- `void replace (const unsigned int start, const unsigned int end, unsigned char after)`
对寄存器值进行部分值的替换操作 例: `replace(7, 4, 0b1111)`; 使寄存器的高四位为1
- `auto operator[] (const size_t position)`
- `template<typename Type >`
`operator Type () const`

3.15.1 详细描述

Register类，提供方便的位操作函数

3.15.2 构造及析构函数说明

Register()

```
template<typename T = int>
constexpr CX9261A::Register::Register (
    const T value = 0 ) [inline], [constexpr]
```

3.15.3 成员函数说明

operator Type()

```
template<typename Type >
CX9261A::Register::operator Type ( ) const [inline]
```

operator[]()

```
auto CX9261A::Register::operator[] (
    const size_t position ) [inline]
```

replace()

```
void CX9261A::Register::replace (
    const unsigned int start,
    const unsigned int end,
    unsigned char after ) [inline]
```

对寄存器值进行部分值的替换操作 例: `replace(7, 4, 0b1111)`; 使寄存器的高四位为1

参数

<i>start</i>	起始位
<i>end</i>	中止位
<i>after</i>	要替换的值

value()

```
unsigned char CX9261A::Register::value ( ) const [inline]
```

以十进制返回寄存器值

3.16 CX9261A::RxConfig结构体 参考

配置接收通道相关参数结构体

```
#include <defs.h>
```

Public 属性

- double [bandwidth](#)
接收通道 *1dB*带宽，单位*MHz*
- int [tia](#)
接收通道 *TIA*增益配置，推荐配置*6*
- int [mixer](#)
接收通道 *Mixer*增益配置

3.16.1 详细描述

配置接收通道相关参数结构体

3.16.2 类成员变量说明

bandwidth

```
double CX9261A::RxConfig::bandwidth
```

接收通道*1dB*带宽，单位*MHz*

mixer

```
int CX9261A::RxConfig::mixer
```

接收通道*Mixer*增益配置

tia

```
int CX9261A::RxConfig::tia
```

接收通道TIA增益配置，推荐配置6

3.17 CX9261A::RxNCO结构体 参考

配置RX NCO相关参数结构体

```
#include <defs.h>
```

Public 属性

- bool [rxUseBist](#)
是否开启*BIST*模式
- bool [rxNCOBypass](#)
是否*Bypass NCO*
- bool [rxNCOUpward](#)
开启上/下混频模式
- bool [rxNCOMaster](#)
使用*Master/Slaver*模式
- double [rxNCOFreq](#)
*NCO*频率

3.17.1 详细描述

配置RX NCO相关参数结构体

3.17.2 类成员变量说明

rxNCOBypass

```
bool CX9261A::RxNCO::rxNCOBypass
```

是否*Bypass NCO*

rxNCOFreq

```
double CX9261A::RxNCO::rxNCOFreq
```

*NCO*频率

rxNCOMaster

```
bool CX9261A::RxNCO::rxNCOMaster
```

使用*Master/Slaver*模式

rxNCOUpward

```
bool CX9261A::RxNCO::rxNCOUpward
```

开启上/下混频模式

rxUseBist

```
bool CX9261A::RxNCO::rxUseBist
```

是否开启BIST模式

3.18 CX9261A::TxConfig结构体 参考

配置发射通道相关参数结构体

```
#include <defs.h>
```

Public 属性

- double **bandwidth**
发射通道1dB带宽，单位MHz
- int **mixer**
发射通道Mixer增益配置

3.18.1 详细描述

配置发射通道相关参数结构体

3.18.2 类成员变量说明**bandwidth**

```
double CX9261A::TxConfig::bandwidth
```

发射通道1dB带宽，单位MHz

mixer

```
int CX9261A::TxConfig::mixer
```

发射通道Mixer增益配置

3.19 CX9261A::TxNCO结构体 参考

配置TX NCO相关参数结构体

```
#include <defs.h>
```

Public 属性

- bool [txUseBist](#)
是否开启*BIST*模式
- double [txNCOFreq](#)
*NCO*频率

3.19.1 详细描述

配置TX NCO相关参数结构体

3.19.2 类成员变量说明

txNCOFreq

```
double CX9261A::TxNCO::txNCOFreq
```

*NCO*频率

txUseBist

```
bool CX9261A::TxNCO::txUseBist
```

是否开启*BIST*模式

4 文件说明

4.1 ChangeLog.txt 文件参考

4.2 cx9261adriver.cpp 文件参考

```
#include "cx9261adriver.h"  
#include "utils.h"
```

命名空间

- namespace [CX9261A](#)

函数

- void `CX9261A::initialChip ()`
初始化芯片状态，写入与具体配置无关的寄存器值
- void `CX9261A::initialPowerConfig (const CX9261AConfiguration &config)`
初始化电源配置，如果只自定义配置部分模式下的电源管理，在自定义配置之前需要先调用此函数
- void `CX9261A::initialBBPLL ()`
初始化BBPLL相关寄存器，如果调用了`initialChip`，则无需单独调用此函数
- void `CX9261A::initialPLL1 ()`
初始化PLL1相关寄存器，如果调用了`initialChip`，则无需单独调用此函数
- void `CX9261A::initialPLL2 ()`
初始化PLL2相关寄存器，如果调用了`initialChip`，则无需单独调用此函数
- void `CX9261A::initialPLL3 ()`
初始化PLL3相关寄存器，如果调用了`initialChip`，则无需单独调用此函数
- void `CX9261A::initialPLL4 ()`
初始化PLL4相关寄存器，如果调用了`initialChip`，则无需单独调用此函数
- void `CX9261A::initialPLL5 ()`
初始化PLL5相关寄存器，如果调用了`initialChip`，则无需单独调用此函数
- void `CX9261A::configRef (double ref)`
配置受参考时钟频率影响的寄存器
- void `CX9261A::configBBPLL (const CX9261AConfiguration &config)`
配置时钟相关寄存器，包括ADC和BBPLL
- bool `CX9261A::afcBBPLL ()`
BBPLL AFC
- std::vector< int > `CX9261A::configPllImpl (const double lo, const double ref)`
- void `CX9261A::configPll1 (double lo, const Clock &clock)`
配置PLL1频率
- bool `CX9261A::afcPll1 ()`
Pll1 AFC
- void `CX9261A::configPll2 (double lo, const Clock &clock)`
配置PLL2频率
- bool `CX9261A::afcPll2 ()`
Pll2 AFC
- void `CX9261A::configPll3 (double lo, const Clock &clock)`
配置PLL3频率
- bool `CX9261A::afcPll3 ()`
Pll3 AFC
- void `CX9261A::configPll4 (double lo, const Clock &clock)`
配置PLL4频率
- bool `CX9261A::afcPll4 ()`
Pll4 AFC
- void `CX9261A::configPll5 (double lo, const Clock &clock)`
配置PLL5频率
- bool `CX9261A::afcPll5 ()`
Pll5 AFC
- void `CX9261A::configAGCMode (int mode)`
配置AGC模式
- void `CX9261A::configAGCPDT (const AGCPDT &pdt)`
配置AGC PDT
- void `CX9261A::configAGCLMT (const AGCLMT &lmt)`
配置AGC LMT

- void `CX9261A::configAGCDIG` (const `AGCDIG` &dig)
配置AGC DIG
- void `CX9261A::configAGC` (const `AGC` &agc)
配置AGC
- void `CX9261A::configRISCV` (bool on)
配置片内处理器，仅在使用需要处理器辅助的校准方法时需要开启处理器
- void `CX9261A::configTemperatureCompensation` (`TemperatureComp` mode)
配置温补
- void `CX9261A::configADCCalib` (double adc)
ADC校准
- void `CX9261A::configFlashCalib` ()
Flash校准
- std::map< int, int > `CX9261A::beforeQecState` ()
QEC校准会改变部分寄存器配置，在进行校准之前需要保存这部分寄存器，校准后恢复
- void `CX9261A::restoreQecState` (const std::map< int, int > &state)
恢复QEC校准过程中改变的寄存器
- void `CX9261A::preparePowerForRxCalib` (const `PllSource` &pllSource)
根据pll来源配置校准模式下的电源管理
- void `CX9261A::preparePllForRxCalib` (const `CX9261AConfiguration` &config, `RxChannel` chan)
调整校准时Pll的配置
- bool `CX9261A::waitRxTSTFinished` ()
等待RxTST窄带校准结束，可重写
- void `CX9261A::rxTSTCalib` (`RxChannel` chan)
RX时域统计校准
- void `CX9261A::rxDelayCalib` (double fs, `RxChannel` chan)
RX延时线校准
- void `CX9261A::configDcCalib` (const `Calib` &config)
DC Tracking
- void `CX9261A::configDcCalibMethod` (const `DCCalibMethod` method)
- void `CX9261A::configDcTrackingLength` (const int length)
- void `CX9261A::configDcHighPassFactor` (const int factor)
- void `CX9261A::rxCalib` (const `CX9261AConfiguration` &config, `RxChannel` chan)
Rx校准
- void `CX9261A::configCalibMethod` (const `Calib` &calib)
选择校准方法
- `CalibPredis` `CX9261A::readRxCalibPredis` (`RxChannel` chan)
读取当前Rx chan通道的校准值
- void `CX9261A::saveRxCalibPredis` (unsigned char index, `RxChannel` chan, `LoSource` pll)
储存校准值
- void `CX9261A::loadRxCalibPredis` (unsigned char index, `RxChannel` chan, `LoSource` pll)
加载保存的校准值
- void `CX9261A::configRxMultiPointsQecImpl` (const `CX9261AConfiguration` &config, const std::vector< double > &points, const `RxChannel` chan)
- void `CX9261A::configRxMultiPointsQec` (const `CX9261AConfiguration` &config, const std::vector< double > &points, `RxChannel` chan)
Rx窄带多点校准
- void `CX9261A::preparePowerForTxCalib` (const `PllSource` &pllSource)
根据pll来源配置校准模式下的电源管理
- void `CX9261A::preparePllForTxCalib` (const `CX9261AConfiguration` &config)
调整校准时Pll的配置
- void `CX9261A::txBisectionQecCalib` (double f0, double fs)

- TX二分法QEC校准
- void `CX9261A::txTSTCalib` (const `CX9261AConfiguration` &config, `TxChannel` chan)
 - TX时域统计法校准
- void `CX9261A::txBisectionDcCalib` (double f0, double fs)
 - TX二分法DC校准
- void `CX9261A::configTxMultiPointsQec` (const `CX9261AConfiguration` &config, const std::vector< double > &points)
 - Tx多点校准
- void `CX9261A::txCalib` (const `CX9261AConfiguration` &config)
 - Tx通道校准
- void `CX9261A::saveTxCalibPredis` (unsigned char index, `TxChannel` chan)
 - 储存校准值
- void `CX9261A::loadTxCalibPredis` (unsigned char index, `TxChannel` chan, `LoSource` pll)
 - 加载保存的校准值
- void `CX9261A::configQecCalib` (const `CX9261AConfiguration` &config)
 - Qec校准顶层函数，内部根据配置选择使用对应方法进行校准
- void `CX9261A::configTxNco` (const `TxNCO` &nco, double fs)
 - 配置TX NCO相关寄存器
- void `CX9261A::configTxBIST` (bool bist)
 - 配置TX NCO BIST模式
- void `CX9261A::configTxNcoFreq` (double freq, double fs)
 - 配置TX NCO频率
- void `CX9261A::configRxNco` (const `RxNCO` &nco, double fs)
 - 配置RX NCO相关寄存器
- void `CX9261A::configRxNcoMode` (`NCOMode` mode)
 - 配置Rx NCO模式
- void `CX9261A::configRxNcoMS` (`NCOMode` mode)
 - 配置Rx NCO模式
- void `CX9261A::configRxNcoFreq` (double freq, double fs)
 - 配置Rx NCO频率
- void `CX9261A::configInterface` (const `Interface` &interface)
 - 配置接口相关寄存器
- void `CX9261A::configPllSource` (const `PllSource` &pllSource)
 - 配置PLL来源
- void `CX9261A::configPowerMode` (`PowerMode` pm)
 - 切换电源模式
- void `CX9261A::configPowerPins` (const `Power` &power, `PowerMode` pm)
 - 配置电源管理
- void `CX9261A::configRefDoubler` (bool doubler, double ref)
 - 配置参考倍频器
- void `CX9261A::initialRxChannel` (`RxChannel` chan)
 - 初始化通道，调用`initialChip`函数会初始化RX1, RX2, RX3三个通道
- std::pair< int, int > `CX9261A::configBandwidthImpl` (const double bandwidth)
- void `CX9261A::configRxBandwidth` (double bandwidth, `RxChannel` chan)
 - 配置Rx带宽
- void `CX9261A::configRxTIA` (int tia, `RxChannel` chan)
 - 配置Rx TIA
- void `CX9261A::configRxMixer` (int mixer)
 - 配置Rx Mixer
- void `CX9261A::configRxChannel` (const `RxConfig` &config, `RxChannel` chan)
 - 配置Rx通道，包含TIA, Mixer以及Bandwidth

- void `CX9261A::configRxFIR` (int fir)
配置 *Rx FIR* 滤波器抽取倍数
- void `CX9261A::configRxHBF1` (int hbf)
配置 *Rx HBF1* 滤波器抽取倍数
- void `CX9261A::configRxHBF2` (int hbf)
配置 *Rx HBF2* 滤波器抽取倍数
- void `CX9261A::configRxHBF3` (int hbf)
配置 *Rx HBF3* 滤波器抽取倍数
- void `CX9261A::configRxCIC` (int cic)
配置 *Rx CIC* 滤波器
- void `CX9261A::configRxFilters` (const `Filters` &filters)
配置 *Rx* 滤波器
- void `CX9261A::configRxFIRCoe` (const std::vector< double > &coe)
配置 *Rx FIR* 滤波器系数
- std::pair< int, int > `CX9261A::configTxBandwidthImpl` (const double bandwidth)
- void `CX9261A::configTxBandwidth` (double bandwidth, `TxChannel` chan)
配置 *Tx* 带宽
- void `CX9261A::configTxMixer` (int mixer, `TxChannel` chan)
配置 *Tx Mixer*
- void `CX9261A::initialTxChannel` (`TxChannel` chan)
初始化 *Tx* 通道
- void `CX9261A::configTxChannelFrequency` (double lo, `TxChannel` chan)
根据 *Tx* 通道的本振频率，配置对应的参数
- void `CX9261A::configTxChannel` (const `TxConfig` &config, `TxChannel` chan)
配置 *Tx* 通道，*Mixer* 以及 *Bandwidth*
- void `CX9261A::configTxFIR` (int fir)
配置 *Tx FIR* 滤波器插值倍数
- void `CX9261A::configTxHBF1` (int hbf)
配置 *Tx HBF1* 滤波器插值倍数
- void `CX9261A::configTxHBF2` (int hbf)
配置 *Tx HBF2* 滤波器插值倍数
- void `CX9261A::configTxHBF3` (int hbf)
配置 *Tx HBF3* 滤波器插值倍数
- void `CX9261A::configTxCIC` (int cic)
配置 *Tx CIC* 滤波器
- void `CX9261A::configTxFilters` (const `Filters` &filters)
配置 *Tx* 滤波器
- void `CX9261A::configTxFIRCoe` (const std::vector< double > &coe)
配置 *Tx FIR* 滤波器系数
- `CalibPredis` `CX9261A::readTxCalibPredis` (`TxChannel` chan)
读取当前 *Tx chan* 通道的校准值
- void `CX9261A::configFPGA` (const `Interface` &interface)
此函数为 *Demo* 板配套配置函数，用于配置 *FPGA* 接口，用户开发自己的程序时应重写对应功能
- void `CX9261A::rxFFTCalib` (const double lo, const double fs, const int firLength, const `CX9261AConfiguration` &config, `RxChannel` chan)
- void `CX9261A::txTSTPhaseDelay` (const `Clock` &clock, const double lo, const double fs, const double f0, const `TxChannel` chan)

4.3 cx9261adriver.h 文件参考

```
#include "defs.h"
```

命名空间

- namespace [CX9261A](#)

宏定义

- #define [VERSION](#) 1.7.0
- #define [LOGGING](#)(x)

函数

- void [CX9261A::writeSpi](#) (int address, int value)
需要实现的函数，实现SPI写入寄存器功能
- int [CX9261A::readSpi](#) (int address)
需要实现的函数，实现SPI读取寄存器功能
- void [CX9261A::pause](#) (double ms)
需要实现的函数，实现等待功能
- void [CX9261A::setPin](#) (int pin, int value)
需要实现的函数，实现配置FPGA对应寄存器功能
- void [CX9261A::initialChip](#) ()
初始化芯片状态，写入与具体配置无关的寄存器值
- void [CX9261A::initialPowerConfig](#) (const [CX9261AConfiguration](#) &config)
初始化电源配置，如果只自定义配置部分模式下的电源管理，在自定义配置之前需要先调用此函数
- void [CX9261A::initialBBPLL](#) ()
初始化BBPLL相关寄存器，如果调用了initialChip，则无需单独调用此函数
- void [CX9261A::initialPLL1](#) ()
初始化PLL1相关寄存器，如果调用了initialChip，则无需单独调用此函数
- void [CX9261A::initialPLL2](#) ()
初始化PLL2相关寄存器，如果调用了initialChip，则无需单独调用此函数
- void [CX9261A::initialPLL3](#) ()
初始化PLL3相关寄存器，如果调用了initialChip，则无需单独调用此函数
- void [CX9261A::initialPLL4](#) ()
初始化PLL4相关寄存器，如果调用了initialChip，则无需单独调用此函数
- void [CX9261A::initialPLL5](#) ()
初始化PLL5相关寄存器，如果调用了initialChip，则无需单独调用此函数
- void [CX9261A::configRef](#) (double ref)
配置受参考时钟频率影响的寄存器
- void [CX9261A::configBBPLL](#) (const [CX9261AConfiguration](#) &config)
配置时钟相关寄存器，包括ADC和BBPLL
- bool [CX9261A::afcBBPLL](#) ()
BBPLL AFC
- void [CX9261A::configPII1](#) (double lo, const [Clock](#) &clock)
配置PLL1频率
- bool [CX9261A::afcPII1](#) ()
PII1 AFC
- void [CX9261A::configPII2](#) (double lo, const [Clock](#) &clock)
配置PLL2频率
- bool [CX9261A::afcPII2](#) ()
PII2 AFC
- void [CX9261A::configPII3](#) (double lo, const [Clock](#) &clock)

- 配置PLL3频率
- bool `CX9261A::afcPll3` ()
 - Pll3 AFC*
- void `CX9261A::configPll4` (double lo, const `Clock` &clock)
 - 配置PLL4频率
- bool `CX9261A::afcPll4` ()
 - Pll4 AFC*
- void `CX9261A::configPll5` (double lo, const `Clock` &clock)
 - 配置PLL5频率
- bool `CX9261A::afcPll5` ()
 - Pll5 AFC*
- void `CX9261A::configAGCMode` (int mode)
 - 配置AGC模式
- void `CX9261A::configAGCPDT` (const `AGCPDT` &pdt)
 - 配置AGC PDT
- void `CX9261A::configAGCLMT` (const `AGCLMT` &lmt)
 - 配置AGC LMT
- void `CX9261A::configAGCDIG` (const `AGCDIG` &dig)
 - 配置AGC DIG
- void `CX9261A::configAGC` (const `AGC` &agc)
 - 配置AGC
- void `CX9261A::configRISCV` (bool on)
 - 配置片内处理器，仅在使用需要处理器辅助的校准方法时需要开启处理器
- void `CX9261A::configTemperatureCompensation` (`TemperatureComp` mode)
 - 配置温补
- void `CX9261A::configADCCalib` (double adc)
 - ADC校准
- void `CX9261A::configFlashCalib` ()
 - Flash校准
- std::map< int, int > `CX9261A::beforeQecState` ()
 - QEC校准会改变部分寄存器配置，在进行校准之前需要保存这部分寄存器，校准后恢复
- void `CX9261A::restoreQecState` (const std::map< int, int > &state)
 - 恢复QEC校准过程中改变的寄存器
- void `CX9261A::preparePowerForRxCalib` (const `PllSource` &pllSource)
 - 根据pll来源配置校准模式下的电源管理
- void `CX9261A::preparePllForRxCalib` (const `CX9261AConfiguration` &config, `RxChannel` chan)
 - 调整校准时Pll的配置
- bool `CX9261A::waitRxTSTFinished` ()
 - 等待RxTST窄带校准结束，可重写
- void `CX9261A::rxTSTCalib` (`RxChannel` chan)
 - RX时域统计校准
- void `CX9261A::rxDelayCalib` (double fs, `RxChannel` chan)
 - RX延时线校准
- void `CX9261A::configDcCalib` (const `Calib` &config)
 - DC Tracking
- void `CX9261A::configDcCalibMethod` (const `DCCalibMethod` method)
- void `CX9261A::configDcTrackingLength` (const int length)
- void `CX9261A::configDcHighPassFactor` (const int factor)
- void `CX9261A::rxCalib` (const `CX9261AConfiguration` &config, `RxChannel` chan)
 - Rx校准
- void `CX9261A::configCalibMethod` (const `Calib` &calib)

- 选择校准方法
- `CalibPredis CX9261A::readRxCalibPredis (RxChannel chan)`
读取当前 *Rx chan* 通道的校准值
- `void CX9261A::saveRxCalibPredis (unsigned char index, RxChannel chan, LoSource pll)`
储存校准值
- `void CX9261A::loadRxCalibPredis (unsigned char index, RxChannel chan, LoSource pll)`
加载保存的校准值
- `void CX9261A::configRxMultiPointsQec (const CX9261AConfiguration &config, const std::vector< double > &points, RxChannel chan)`
*Rx*窄带多点校准
- `void CX9261A::preparePowerForTxCalib (const PllSource &pllSource)`
根据 *pll* 来源配置校准模式下的电源管理
- `void CX9261A::preparePllForTxCalib (const CX9261AConfiguration &config)`
调整校准时 *PLL* 的配置
- `void CX9261A::txBisectionQecCalib (double f0, double fs)`
*TX*二分法 *QEC* 校准
- `void CX9261A::txTSTCalib (const CX9261AConfiguration &config, TxChannel chan)`
*TX*时域统计法校准
- `void CX9261A::txBisectionDcCalib (double f0, double fs)`
*TX*二分法 *DC* 校准
- `void CX9261A::configTxMultiPointsQec (const CX9261AConfiguration &config, const std::vector< double > &points)`
*Tx*多点校准
- `void CX9261A::txCalib (const CX9261AConfiguration &config)`
*Tx*通道校准
- `void CX9261A::saveTxCalibPredis (unsigned char index, TxChannel chan)`
储存校准值
- `void CX9261A::loadTxCalibPredis (unsigned char index, TxChannel chan, LoSource pll)`
加载保存的校准值
- `void CX9261A::configQecCalib (const CX9261AConfiguration &config)`
Qec 校准顶层函数，内部根据配置选择使用对应方法进行校准
- `void CX9261A::configTxNco (const TxNCO &nco, double fs)`
配置 *TX NCO* 相关寄存器
- `void CX9261A::configTxBIST (bool bist)`
配置 *TX NCO BIST* 模式
- `void CX9261A::configTxNcoFreq (double freq, double fs)`
配置 *TX NCO* 频率
- `void CX9261A::configRxNco (const RxNCO &nco, double fs)`
配置 *RX NCO* 相关寄存器
- `void CX9261A::configRxNcoMode (NCOMode mode)`
配置 *Rx NCO* 模式
- `void CX9261A::configRxNcoMS (NCOMode mode)`
配置 *Rx NCO* 模式
- `void CX9261A::configRxNcoFreq (double freq, double fs)`
配置 *Rx NCO* 频率
- `void CX9261A::configInterface (const Interface &interface)`
配置接口相关寄存器
- `void CX9261A::configPllSource (const PllSource &pllSource)`
配置 *PLL* 来源
- `void CX9261A::configPowerMode (PowerMode pm)`
切换电源模式

- void `CX9261A::configPowerPins` (const `Power` &power, `PowerMode` pm)
配置电源管理
- void `CX9261A::configRefDoubler` (bool doubler, double ref)
配置参考倍频器
- void `CX9261A::initialRxChannel` (`RxChannel` chan)
初始化通道, 调用`initialChip`函数会初始化`RX1`, `RX2`, `RX3`三个通道
- void `CX9261A::configRxBandwidth` (double bandwidth, `RxChannel` chan)
配置`Rx`带宽
- void `CX9261A::configRxTIA` (int tia, `RxChannel` chan)
配置`Rx TIA`
- void `CX9261A::configRxMixer` (int mixer)
配置`Rx Mixer`
- void `CX9261A::configRxChannel` (const `RxConfig` &config, `RxChannel` chan)
配置`Rx`通道, 包含`TIA`, `Mixer`以及`Bandwidth`
- void `CX9261A::configRxFIR` (int fir)
配置`Rx FIR`滤波器抽取倍数
- void `CX9261A::configRxHBF1` (int hbf)
配置`Rx HBF1`滤波器抽取倍数
- void `CX9261A::configRxHBF2` (int hbf)
配置`Rx HBF2`滤波器抽取倍数
- void `CX9261A::configRxHBF3` (int hbf)
配置`Rx HBF3`滤波器抽取倍数
- void `CX9261A::configRxCIC` (int cic)
配置`Rx CIC`滤波器
- void `CX9261A::configRxFilters` (const `Filters` &filters)
配置`Rx`滤波器
- void `CX9261A::configRxFIRCoe` (const std::vector< double > &coe)
配置`Rx FIR`滤波器系数
- void `CX9261A::configTxBandwidth` (double bandwidth, `TxChannel` chan)
配置`Tx`带宽
- void `CX9261A::configTxMixer` (int mixer, `TxChannel` chan)
配置`Tx Mixer`
- void `CX9261A::initialTxChannel` (`TxChannel` chan)
初始化`Tx`通道
- void `CX9261A::configTxChannelFrequency` (double lo, `TxChannel` chan)
根据`Tx`通道的本振频率, 配置对应的参数
- void `CX9261A::configTxChannel` (const `TxConfig` &config, `TxChannel` chan)
配置`Tx`通道, `Mixer`以及`Bandwidth`
- void `CX9261A::configTxFIR` (int fir)
配置`Tx FIR`滤波器插值倍数
- void `CX9261A::configTxHBF1` (int hbf)
配置`Tx HBF1`滤波器插值倍数
- void `CX9261A::configTxHBF2` (int hbf)
配置`Tx HBF2`滤波器插值倍数
- void `CX9261A::configTxHBF3` (int hbf)
配置`Tx HBF3`滤波器插值倍数
- void `CX9261A::configTxCIC` (int cic)
配置`Tx CIC`滤波器
- void `CX9261A::configTxFilters` (const `Filters` &filters)
配置`Tx`滤波器
- void `CX9261A::configTxFIRCoe` (const std::vector< double > &coe)

配置Tx FIR滤波器系数

- [CalibPredis CX9261A::readTxCalibPredis \(TxChannel chan\)](#)

读取当前Tx chan通道的校准值

- void [CX9261A::configFPGA \(const Interface &interface\)](#)

此函数为Demo板配套配置函数，用于配置FPGA接口，用户开发自己的程序时应重写对应功能

4.3.1 宏定义说明

LOGGING

```
#define LOGGING(  
    x )
```

VERSION

```
#define VERSION 1.7.0
```

4.4 cx9261adriver.h

[浏览该文件的文档.](#)

```
00001  
00005 #ifndef CX9261ADriver_CX9261ADriver_H  
00006 #define CX9261ADriver_CX9261ADriver_H  
00007  
00008 #define VERSION 1.7.0  
00009  
00010 #include "defs.h"  
00011 // 如果不需要输出日志内容，注释掉下面的宏定义  
00012 // #define USE_LOGGING  
00013  
00014 #ifdef USE_LOGGING  
00015 #    define LOGGING(x) logging(x);  
00016 #else  
00017 #    define LOGGING(x)  
00018 #endif  
00019  
00020 namespace CX9261A {  
00021 #ifdef USE_LOGGING  
00026 void logging(const std::string& msg);  
00027 #endif  
00033 void writeSpi(int address, int value);  
00039 int readSpi(int address);  
00044 void pause(double ms);  
00050 void setPin(int pin, int value);  
00054 void initialChip();  
00058 void initialPowerConfig(const CX9261AConfiguration& config);  
00062 void initialBBPLL();  
00066 void initialPLL1();  
00070 void initialPLL2();  
00074 void initialPLL3();  
00078 void initialPLL4();  
00082 void initialPLL5();  
00087 void configRef(double ref);  
00092 void configBBPLL(const CX9261AConfiguration& config);  
00097 bool afcBBPll();  
00103 void configPll1(double lo, const Clock& clock);  
00108 bool afcPll1();  
00114 void configPll2(double lo, const Clock& clock);  
00119 bool afcPll2();  
00125 void configPll3(double lo, const Clock& clock);  
00130 bool afcPll3();  
00136 void configPll4(double lo, const Clock& clock);  
00141 bool afcPll4();  
00147 void configPll5(double lo, const Clock& clock);  
00152 bool afcPll5();  
00157 void configAGCMode(int mode);  
00162 void configAGCPDT(const AGCPDT& pdt);  
00167 void configAGCLMT(const AGCLMT& lmt);
```

```

00172 void configAGCDIG(const AGCDIG& dig);
00177 void configAGC(const AGC& agc);
00182 void configRISCV(bool on);
00187 void configTemperatureCompensation(TemperatureComp mode);
00192 void configADCCalib(double adc);
00196 void configFlashCalib();
00201 std::map<int, int> beforeQecState();
00206 void restoreQecState(const std::map<int, int>& state);
00211 void preparePowerForRxCalib(const PllSource& pllSource);
00217 void preparePllForRxCalib(const CX9261AConfiguration& config, RxChannel chan);
00222 bool waitRxTSTFinished();
00227 void rxTSTCalib(RxChannel chan);
00233 void rxDelayCalib(double fs, RxChannel chan);
00238 void configDcCalib(const Calib& config);
00239 void configDcCalibMethod(DCCalibMethod method);
00240 void configDcTrackingLength(int length);
00241 void configDcHighPassFactor(int factor);
00247 void rxCalib(const CX9261AConfiguration& config, RxChannel chan);
00252 void configCalibMethod(const Calib& calib);
00258 CalibPredis readRxCalibPredis(RxChannel chan);
00265 void saveRxCalibPredis(unsigned char index, RxChannel chan, LoSource pll);
00272 void loadRxCalibPredis(unsigned char index, RxChannel chan, LoSource pll);
00279 void configRxMultiPointsQec(const CX9261AConfiguration& config, const std::vector<double>& points,
    RxChannel chan);
00284 void preparePowerForTxCalib(const PllSource& pllSource);
00289 void preparePllForTxCalib(const CX9261AConfiguration& config);
00295 void txBisectionQecCalib(double f0, double fs);
00301 void txTSTCalib(const CX9261AConfiguration& config, TxChannel chan);
00307 void txBisectionDcCalib(double f0, double fs);
00313 void configTxMultiPointsQec(const CX9261AConfiguration& config, const std::vector<double>& points);
00318 void txCalib(const CX9261AConfiguration& config);
00324 void saveTxCalibPredis(unsigned char index, TxChannel chan);
00331 void loadTxCalibPredis(unsigned char index, TxChannel chan, LoSource pll);
00336 void configQecCalib(const CX9261AConfiguration& config);
00342 void configTxNco(const TxNCO& nco, double fs);
00347 void configTxBIST(bool bist);
00353 void configTxNcoFreq(double freq, double fs);
00359 void configRxNco(const RxNCO& nco, double fs);
00364 void configRxNcoMode(NCOMode mode);
00369 void configRxNcoMS(NCOMode mode);
00375 void configRxNcoFreq(double freq, double fs);
00380 void configInterface(const Interface& interface);
00385 void configPllSource(const PllSource& pllSource);
00390 void configPowerMode(PowerMode pm);
00396 void configPowerPins(const Power& power, PowerMode pm);
00402 void configRefDoubler(bool doubler, double ref);
00407 void initialRxChannel(RxChannel chan);
00414 void configRxBandwidth(double bandwidth, RxChannel chan);
00421 void configRxTIA(int tia, RxChannel chan);
00426 void configRxMixer(int mixer);
00432 void configRxChannel(const RxConfig& config, RxChannel chan);
00437 void configRxFIR(int fir);
00442 void configRxHBF1(int hbf);
00447 void configRxHBF2(int hbf);
00452 void configRxHBF3(int hbf);
00457 void configRxCIC(int cic);
00462 void configRxFilters(const Filters& filters);
00467 void configRxFIRCoef(const std::vector<double>& coe);
00473 void configTxBandwidth(double bandwidth, TxChannel chan);
00479 void configTxMixer(int mixer, TxChannel chan);
00484 void initialTxChannel(TxChannel chan);
00490 void configTxChannelFrequency(double lo, TxChannel chan);
00496 void configTxChannel(const TxConfig& config, TxChannel chan);
00501 void configTxFIR(int fir);
00506 void configTxHBF1(int hbf);
00511 void configTxHBF2(int hbf);
00516 void configTxHBF3(int hbf);
00521 void configTxCIC(int cic);
00526 void configTxFilters(const Filters& filters);
00531 void configTxFIRCoef(const std::vector<double>& coe);
00537 CalibPredis readTxCalibPredis(TxChannel chan);
00542 void configFPGA(const Interface& interface);
00543 } // namespace CX9261A
00544
00545 #endif // CX9261ADRIVER_CX9261ADRIVER_H

```

4.5 defs.h 文件参考

```

#include <bitset>
#include <map>
#include <vector>

```

类

- struct [CX9261A::AGCPDT](#)
[AGC PDT](#)模式的结构体
- struct [CX9261A::AGCLMT](#)
[AGC LMT](#)模式的结构体
- struct [CX9261A::AGCDIG](#)
[AGC DIG](#)模式的结构体
- struct [CX9261A::AGC](#)
[AGC](#)模式的结构体
- struct [CX9261A::Clock](#)
时钟相关参数结构体
- struct [CX9261A::Calib](#)
校准相关参数结构体
- struct [CX9261A::PllSource](#)
配置 PLL 来源相关参数结构体
- struct [CX9261A::PllFreq](#)
配置 PLL 频率相关参数结构体
- struct [CX9261A::Interface](#)
配置接口相关参数结构体
- struct [CX9261A::TxNCO](#)
配置 $TX\ NCO$ 相关参数结构体
- struct [CX9261A::RxnCO](#)
配置 $RX\ NCO$ 相关参数结构体
- struct [CX9261A::RxConfig](#)
配置接收通道相关参数结构体
- struct [CX9261A::TxConfig](#)
配置发射通道相关参数结构体
- struct [CX9261A::Filters](#)
配置滤波器相关参数结构体, 0代表bypass
- struct [CX9261A::Power](#)
配置电源模式结构体, 1代表该模块上电
- struct [CX9261A::PowerManagement](#)
配置电源模式管理, 包含模式选择, 接收模式, 发射模式, FDD 模式
- struct [CX9261A::CalibPredis](#)
储存窄带校准结果的结构体
- struct [CX9261A::CX9261AConfiguration](#)
 $CX9261A$ 结构体

命名空间

- namespace [CX9261A](#)

枚举

- enum class `CX9261A::WorkMode` : `std::uint8_t` { `CX9261A::Rx` , `CX9261A::Tx` , `CX9261A::FDD` }
芯片工作模式
- enum class `CX9261A::TrMode` : `std::uint8_t` { `CX9261A::R1T1` , `CX9261A::R2T2` }
收发数据通道数量
- enum class `CX9261A::PortMode` : `std::uint8_t` { `CX9261A::Single` , `CX9261A::Dual` }
收发接口通道数量
- enum class `CX9261A::IOMode` : `std::uint8_t` { `CX9261A::CMOS` , `CX9261A::LVDS_S` , `CX9261A::LVDS_E` }
接口模式: *CMOS CMOS*接口模式, *LVDS_S LVDS*顺序模式, *LVDS_E LVDS*奇偶模式
- enum class `CX9261A::LoSource` : `std::uint8_t` {
`CX9261A::PII1` = 0b00001 , `CX9261A::PII2` = 0b00010 , `CX9261A::PII3` = 0b00100 , `CX9261A::PII4` = 0b01000
,
`CX9261A::PII5` = 0b10000 , `CX9261A::Off` = 0 }
本振来源
- enum class `CX9261A::RxChannel` : `std::uint8_t` { `CX9261A::Rx1` = 1 , `CX9261A::Rx2` , `CX9261A::Rx3` }
接收通道
- enum class `CX9261A::TxChannel` : `std::uint8_t` { `CX9261A::Tx1` = 1 , `CX9261A::Tx2` }
发射通道
- enum class `CX9261A::PowerMode` : `std::uint8_t` {
`CX9261A::Shutdown` = 0 , `CX9261A::Sleep` , `CX9261A::Alert` , `CX9261A::Calib` ,
`CX9261A::Receive` , `CX9261A::Transmit` , `CX9261A::Duplex` , `CX9261A::Obs` }
电源模式
- enum class `CX9261A::NCOMode` : `std::uint8_t` {
`CX9261A::Bypass` = 0 , `CX9261A::Up` , `CX9261A::Down` , `CX9261A::BIST` ,
`CX9261A::Master` , `CX9261A::Slaver` }
*NCO*模式
- enum class `CX9261A::TemperatureComp` : `std::uint8_t` { `CX9261A::DAC` = 0 , `CX9261A::Band` }
温补模式 *DAC*: *DAC*模式 *Band*: 跳子带模式
- enum class `CX9261A::CalibMethod` : `std::uint8_t` { `CX9261A::TST` = 0 , `CX9261A::Bisec` , `CX9261A::MultiPoints` , `CX9261A::Tracking` }
校准方法 *TST*: 时域统计 *Rx*通道使用, *Bisec*: 二分法 *Tx*通道使用, *MultiPoints*: 多点校准
- enum class `CX9261A::DCCalibMethod` : `std::uint8_t` { `CX9261A::Tracking` = 0 , `CX9261A::HighPass` }
*DC*校准方法 *Tracking*: 跟踪校准 *HighPass*: 高通滤波器滤掉直流

函数

- `std::string CX9261A::toString` (const `LoSource` source)
- `std::string CX9261A::toString` (const `RxChannel` chan)
- `std::string CX9261A::toString` (const `TxChannel` chan)

变量

- `constexpr double CX9261A::delay` = 100
*AFC*等待延时, 单位`ms`
- `constexpr double CX9261A::qecDelay` = 500
*QEC*等待延时, 单位`ms`

4.6 defs.h

[浏览该文件的文档.](#)

```

00001
00005 #ifndef CX9261ADriver_DEFS_H
00006 #define CX9261ADriver_DEFS_H
00007
00008 #include <bitset>
00009 #include <map>
00010 #include <vector>
00011
00012 #ifdef interface
00013 #    undef interface
00014 #endif
00015
00016 namespace CX9261A {
00020 enum class WorkMode : std::uint8_t { Rx, Tx, FDD };
00025 enum class TrMode : std::uint8_t { R1T1, R2T2 };
00029 enum class PortMode : std::uint8_t { Single, Dual };
00036 enum class IOMode : std::uint8_t { CMOS, LVDS_S, LVDS_E };
00040 enum class LoSource : std::uint8_t {
00041     P111 = 0b00001,
00042     P112 = 0b00010,
00043     P113 = 0b00100,
00044     P114 = 0b01000,
00045     P115 = 0b10000,
00046     Off = 0
00047 };
00048
00049 inline std::string toString(const LoSource source)
00050 {
00051     switch (source) {
00052     case LoSource::P111:
00053         return "P111";
00054     case LoSource::P112:
00055         return "P112";
00056     case LoSource::P113:
00057         return "P113";
00058     case LoSource::P114:
00059         return "P114";
00060     case LoSource::P115:
00061         return "P115";
00062     default:
00063         return "Off";
00064     }
00065 }
00069 enum class RxChannel : std::uint8_t { Rx1 = 1, Rx2, Rx3 };
00070
00071 inline std::string toString(const RxChannel chan)
00072 {
00073     switch (chan) {
00074     case RxChannel::Rx1:
00075         return "Rx1";
00076     case RxChannel::Rx2:
00077         return "Rx2";
00078     case RxChannel::Rx3:
00079         return "Rx3";
00080     default:
00081         return "Unknown Channel";
00082     }
00083 }
00087 enum class TxChannel : std::uint8_t { Tx1 = 1, Tx2 };
00088
00089 inline std::string toString(const TxChannel chan)
00090 {
00091     switch (chan) {
00092     case TxChannel::Tx1:
00093         return "Tx1";
00094     case TxChannel::Tx2:
00095         return "Tx2";
00096     default:
00097         return "Unknown Channel";
00098     }
00099 }
00103 enum class PowerMode : std::uint8_t { Shutdown = 0, Sleep, Alert, Calib, Receive, Transmit, Duplex,
Obs };
00107 enum class NCOMode : std::uint8_t { Bypass = 0, Up, Down, BIST, Master, Slaver };
00111 enum class TemperatureComp : std::uint8_t { DAC = 0, Band };
00115 enum class CalibMethod : std::uint8_t { TST = 0, Bisec, MultiPoints, Tracking };
00119 enum class DCCalibMethod : std::uint8_t { Tracking = 0, HighPass };
00123 constexpr double delay = 100;
00127 constexpr double gecDelay = 500;
00131 static const std::map<std::pair<LoSource, LoSource>, std::bitset<7>> rxSrcMap = {
00132     { { LoSource::P112, LoSource::P112 }, 0b0010000 }, { { LoSource::P111, LoSource::P111 }, 0b1101000
},

```

```

00133     { { LoSource::P112, LoSource::P111 }, 0b0100000 }, { { LoSource::P111, LoSource::P112 }, 0b1000000
00134 }, { { LoSource::Off, LoSource::P111 }, 0b0101010 }, { { LoSource::Off, LoSource::P112 }, 0b1010010
00135 }, { { LoSource::P111, LoSource::Off }, 0b1001001 }, { { LoSource::P112, LoSource::Off }, 0b0110001
00136 }, { { LoSource::Off, LoSource::Off }, 0b0011011 }
00137 };
00141 static const std::map<std::pair<LoSource, LoSource>, std::bitset<11>> txSrcMap = {
00142     { { LoSource::P114, LoSource::P114 }, 16 }, { { LoSource::P113, LoSource::P113 }, 104 },
00143     { { LoSource::P114, LoSource::P113 }, 32 }, { { LoSource::P113, LoSource::P114 }, 64 },
00144     { { LoSource::P112, LoSource::P112 }, 1560 }, { { LoSource::P111, LoSource::P111 }, 1944 },
00145     { { LoSource::P112, LoSource::P111 }, 1688 }, { { LoSource::P111, LoSource::P112 }, 1816 },
00146     { { LoSource::Off, LoSource::P113 }, 42 }, { { LoSource::Off, LoSource::P114 }, 82 },
00147     { { LoSource::Off, LoSource::P112 }, 794 }, { { LoSource::Off, LoSource::P111 }, 666 },
00148     { { LoSource::P113, LoSource::Off }, 73 }, { { LoSource::P114, LoSource::Off }, 49 },
00149     { { LoSource::P112, LoSource::Off }, 1177 }, { { LoSource::P111, LoSource::Off }, 1305 },
00150     { { LoSource::Off, LoSource::Off }, 27 }, { { LoSource::P111, LoSource::P113 }, 1320 },
00151     { { LoSource::P111, LoSource::P114 }, 1360 }, { { LoSource::P112, LoSource::P113 }, 1064 },
00152     { { LoSource::P112, LoSource::P114 }, 1104 }, { { LoSource::P113, LoSource::P111 }, 712 },
00153     { { LoSource::P114, LoSource::P111 }, 688 }, { { LoSource::P113, LoSource::P112 }, 584 },
00154     { { LoSource::P114, LoSource::P112 }, 560 }
00155 };
00159 static const std::vector<std::vector<std::vector<int>>> coreTable = {
00160     { { 0x0C, 0x30, 0x08 }, { 0x0B, 0x30, 0x09 }, { 0x0C, 0x30, 0x05 }, { 0x0D, 0x30, 0x08 } },
00161     { { 0x0C, 0x30, 0x0C }, { 0x0B, 0x30, 0x0E }, { 0x0C, 0x30, 0x08 }, { 0x0D, 0x30, 0x0B } },
00162     { { 0x0C, 0x20, 0x10 }, { 0x0B, 0x20, 0x10 }, { 0x0C, 0x20, 0x0A }, { 0x0D, 0x20, 0x0D } },
00163     { { 0x0C, 0x20, 0x11 }, { 0x0B, 0x20, 0x13 }, { 0x0C, 0x20, 0x0C }, { 0x0D, 0x20, 0x0F } },
00164     { { 0x0D, 0x30, 0x14 }, { 0x0D, 0x22, 0x14 }, { 0x0D, 0x20, 0x0D }, { 0x0D, 0x15, 0x0C } },
00165     { { 0x0E, 0x30, 0x18 }, { 0x0E, 0x25, 0x16 }, { 0x0E, 0x25, 0x0F }, { 0x0E, 0x18, 0x0F } },
00166     { { 0x0F, 0x38, 0x1A }, { 0x0F, 0x32, 0x1C }, { 0x0F, 0x18, 0x12 }, { 0x0F, 0x18, 0x12 } }
00167 };
00168
00172 struct AGCPDT
00173 {
00174     int pdtN;
00175     int pdtM;
00176 };
00177
00181 struct AGCLMT
00182 {
00183     int count;
00184     int threshold1;
00185     int threshold2;
00186     int threshold3;
00187     int threshold4;
00188 };
00189
00193 struct AGCDIG
00194 {
00195     int count;
00196     int threshold1;
00197     int threshold2;
00198     int threshold3;
00199     int threshold4;
00200 };
00201
00205 struct AGC
00206 {
00207     int mode;
00208     AGCPDT pdt;
00209     AGCLMT lmt;
00210     AGCDIG dig;
00211 };
00212
00216 struct Clock
00217 {
00218     double ref;
00219     double adc;
00220     int refDiv;
00221     bool bbUseIntDiv;
00222     bool refDoubler;
00223 };
00224
00228 struct Calib
00229 {
00230     bool calibDC;
00231     DCCalibMethod dcCalibMethod;
00232     int dcTrackingLength;
00233     int dcHighPassFactor;
00234     std::uint8_t rx1CalibP11;
00235     std::uint8_t rx2CalibP11;
00236     std::uint8_t rx3CalibP11;
00237     std::uint8_t tx1CalibP11;
00238     std::uint8_t tx2CalibP11;
00239     CalibMethod rxMethod;
00240     CalibMethod txMethod;

```

```

00241     double f0;
00242     std::vector<double> rxMultiPoints;
00243     std::vector<double> txMultiPoints;
00244 };
00245
00249 struct PllSource
00250 {
00251     bool rxUseExt;
00252     bool txUseExt;
00253     LoSource rx1Source;
00254     LoSource rx2Source;
00255     LoSource rx3Source;
00256     LoSource tx1Source;
00257     LoSource tx2Source;
00258 };
00259
00263 struct PllFreq
00264 {
00265     double pll1Freq;
00266     double pll2Freq;
00267     double pll3Freq;
00268     double pll4Freq;
00269     double pll5Freq;
00270     double rxExtFreq;
00271     double txExtFreq;
00272 };
00273
00277 struct Interface
00278 {
00279     WorkMode workMode;
00280     TrMode trMode;
00281     PortMode portMode;
00282     IOMode ioMode;
00283 };
00284
00288 struct TxNCO
00289 {
00290     bool txUseBist;
00291     double txNCOFreq;
00292 };
00293
00297 struct RxNCO
00298 {
00299     bool rxUseBist;
00300     bool rxNCOBypass;
00301     bool rxNCOUpward;
00302     bool rxNCOMaster;
00303     double rxNCOFreq;
00304 };
00305
00309 struct RxConfig
00310 {
00311     double bandwidth;
00312     int tia;
00313     int mixer;
00314 };
00315
00319 struct TxConfig
00320 {
00321     double bandwidth;
00322     int mixer;
00323 };
00324
00328 struct Filters
00329 {
00330     int fir;
00331     int hbf1;
00332     int hbf2;
00333     int hbf3;
00334     int cic;
00335 };
00336
00340 struct Power
00341 {
00342     bool dig_tx2;
00343     bool dig_tx1;
00344     bool dig_rx3;
00345     bool dig_rx2;
00346     bool dig_rx1;
00347     bool frefl_rx;
00348     bool auxadc2;
00349     bool auxadc1;
00350     bool auxdac;
00351     bool temp_sensor;
00352     bool bgr;
00353     bool exlo;
00354     bool tx2;

```

```

00355     bool tx1;
00356     bool rx3;
00357     bool rx2;
00358     bool rx1;
00359     bool pll5_1do;
00360     bool pll5;
00361     bool pll4_1do;
00362     bool pll4;
00363     bool pll3_1do;
00364     bool pll3;
00365     bool pll2_1do;
00366     bool pll2;
00367     bool pll1_1do;
00368     bool pll1;
00369     bool bbpll;
00370 };
00371
00375 struct PowerManagement
00376 {
00377     PowerMode powerMode;
00378     Power rxPower;
00379     Power txPower;
00380     Power fddPower;
00381 };
00382
00386 struct CalibPredis
00387 {
00388     int bHigh;
00389     int bLow;
00390     int cHigh;
00391     int cLow;
00392     int dciHigh;
00393     int dciLow;
00394     int dcqHigh;
00395     int dcqLow;
00396 };
00397
00401 struct CX9261AConfiguration
00402 {
00403     TemperatureComp tempComp;
00404     AGC agc;
00405     Interface interface;
00406     Clock clock;
00407     PllSource pllSource;
00408     Calib calib;
00409     PllFreq pllFreq;
00410     TxNCO txNco;
00411     RxNCO rxNco;
00412     RxConfig rx1Config;
00413     RxConfig rx2Config;
00414     RxConfig rx3Config;
00415     TxConfig tx1Config;
00416     TxConfig tx2Config;
00417     Filters rxFilters;
00418     Filters txFilters;
00419     PowerManagement powerManagement;
00420     std::vector<double> rxFIRCoef;
00421     std::vector<double> txFIRCoef;
00422 };
00423 } // namespace CX9261A
00424 #endif // CX9261ADRIVER_DEFS_H

```

4.7 readme.txt 文件参考

4.8 utils.h 文件参考

```

#include <algorithm>
#include <bitset>
#include <cmath>
#include <string>

```

类

- class [CX9261A::Register](#)

*Register*类，提供方便的位操作函数

命名空间

- namespace [CX9261A](#)

函数

- `template<typename T>`
`constexpr const T & CX9261A::clamp (const T &v, const T &lo, const T &hi)`
- `template<typename Float>`
`int CX9261A::round (Float f)`
- `template<typename Float>`
`long long CX9261A::round64 (Float f)`
- `template<typename Float>`
`int CX9261A::floor (Float f)`
- `template<typename Float>`
`long long CX9261A::floor64 (Float f)`
- `template<typename Float>`
`int CX9261A::ceil (Float f)`
- `template<typename Float>`
`long long CX9261A::ceil64 (Float f)`
- `template<int N>`
`std::string CX9261A::frac2bin (const double frac)`

4.9 utils.h

[浏览该文件的文档.](#)

```

00001
00005 #ifndef CX9261ADriver_UTILS_H
00006 #define CX9261ADriver_UTILS_H
00007 #include <algorithm>
00008 #include <bitset>
00009 #include <cmath>
00010 #include <string>
00011 namespace CX9261A {
00012 #if __cplusplus >= 201703L
00013 template <typename T>
00014 constexpr const T& clamp(const T& v, const T& lo, const T& hi) {
00015     return std::clamp(v, lo, hi);
00016 }
00017 #else
00018 template <typename T>
00019 constexpr const T& clamp(const T& v, const T& lo, const T& hi) {
00020     return std::max(std::min(hi, v), lo);
00021 }
00022 #endif
00023
00024 template <typename Float>
00025 int round(Float f) {
00026     return static_cast<int>(std::round(f));
00027 }
00028
00029 template <typename Float>
00030 long long round64(Float f) {
00031     return static_cast<long long>(std::round(f));
00032 }
00033
00034 template <typename Float>
00035 int floor(Float f) {
00036     return static_cast<int>(std::floor(f));
00037 }
00038
00039 template <typename Float>
00040 long long floor64(Float f) {
00041     return static_cast<long long>(std::floor(f));
00042 }
00043
00044 template <typename Float>
00045 int ceil(Float f) {
00046     return static_cast<int>(std::ceil(f));

```

```
00047 }
00048
00049 template <typename Float>
00050 long long ceil64(Float f) {
00051     return static_cast<long long>(std::ceil(f));
00052 }
00053
00054 class Register {
00055 public:
00056     template <typename T = int>
00057     constexpr Register(const T value = 0) {
00058         static_assert(std::is_integral_v<T>, "Register value must be integral!");
00059         m_value = static_cast<unsigned long long>(value);
00060     }
00061     unsigned char value() const { return static_cast<unsigned char>(m_value.to_ulong()); }
00062     void replace(const unsigned int start, const unsigned int end, unsigned char after) {
00063         for (auto i = end; i <= start; ++i) {
00064             m_value[i] = (after >> (i - end)) & 0b1;
00065         }
00066     }
00067     auto operator[](const size_t position) { return m_value[position]; }
00068
00069     template <typename Type>
00070     operator Type() const {
00071         static_assert(std::is_integral_v<Type>, "Type must be an integral");
00072         return static_cast<Type>(m_value.to_ulong());
00073     }
00074 private:
00075     std::bitset<8> m_value{};
00076 };
00077
00078 template <int N>
00079 std::string frac2bin(const double frac) {
00080     const long double num = frac * (pow(2.0, N));
00081     std::string ret = std::bitset<N>(std::floor(num)).to_string();
00082     return ret;
00083 }
00084 } // namespace CX9261A
00085 #endif // CX9261ADriverUtils.H
```

Index

adc
 CX9261A::Clock, [46](#)
afcBBPll
 CX9261A, [10](#)
afcPll1
 CX9261A, [10](#)
afcPll2
 CX9261A, [10](#)
afcPll3
 CX9261A, [11](#)
afcPll4
 CX9261A, [11](#)
afcPll5
 CX9261A, [11](#)
agc
 CX9261A::CX9261AConfiguration, [48](#)
Alert
 CX9261A, [9](#)
auxadc1
 CX9261A::Power, [56](#)
auxadc2
 CX9261A::Power, [56](#)
auxdac
 CX9261A::Power, [56](#)

Band
 CX9261A, [9](#)
bandwidth
 CX9261A::RxConfig, [61](#)
 CX9261A::TxConfig, [63](#)
bbpll
 CX9261A::Power, [56](#)
bbUseIntDiv
 CX9261A::Clock, [46](#)
beforeQecState
 CX9261A, [11](#)
bgr
 CX9261A::Power, [56](#)
bHigh
 CX9261A::CalibPredis, [45](#)
Bisec
 CX9261A, [7](#)
BIST
 CX9261A, [8](#)
bLow
 CX9261A::CalibPredis, [45](#)
Bypass
 CX9261A, [8](#)

Calib
 CX9261A, [9](#)
calib
 CX9261A::CX9261AConfiguration, [48](#)
calibDC
 CX9261A::Calib, [43](#)
CalibMethod
 CX9261A, [7](#)
ceil
 CX9261A, [11](#)
ceil64
 CX9261A, [12](#)
ChangeLog.txt, [64](#)
cHigh
 CX9261A::CalibPredis, [45](#)
cic
 CX9261A::Filters, [51](#)
clamp
 CX9261A, [12](#)
clock
 CX9261A::CX9261AConfiguration, [48](#)
cLow
 CX9261A::CalibPredis, [45](#)
CMOS
 CX9261A, [7](#)
configADCCalib
 CX9261A, [12](#)
configAGC
 CX9261A, [12](#)
configAGCDIG
 CX9261A, [12](#)
configAGCLMT
 CX9261A, [13](#)
configAGCMode
 CX9261A, [13](#)
configAGCPDT
 CX9261A, [13](#)
configBandwidthImpl
 CX9261A, [13](#)
configBBPLL
 CX9261A, [13](#)
configCalibMethod
 CX9261A, [14](#)
configDcCalib
 CX9261A, [14](#)
configDcCalibMethod
 CX9261A, [14](#)
configDcHighPassFactor
 CX9261A, [14](#)
configDcTrackingLength
 CX9261A, [14](#)
configFlashCalib
 CX9261A, [15](#)
configFPGA
 CX9261A, [15](#)
configInterface
 CX9261A, [15](#)
configPll1
 CX9261A, [15](#)
configPll2
 CX9261A, [15](#)
configPll3

CX9261A, [16](#)
 configPll4
 CX9261A, [16](#)
 configPll5
 CX9261A, [16](#)
 configPllImpl
 CX9261A, [17](#)
 configPllSource
 CX9261A, [17](#)
 configPowerMode
 CX9261A, [17](#)
 configPowerPins
 CX9261A, [17](#)
 configQecCalib
 CX9261A, [18](#)
 configRef
 CX9261A, [18](#)
 configRefDoubler
 CX9261A, [18](#)
 configRISCV
 CX9261A, [18](#)
 configRxBandwidth
 CX9261A, [18](#)
 configRxChannel
 CX9261A, [19](#)
 configRxCIC
 CX9261A, [19](#)
 configRxFilters
 CX9261A, [19](#)
 configRxFIR
 CX9261A, [20](#)
 configRxFIRCoef
 CX9261A, [20](#)
 configRxHBF1
 CX9261A, [20](#)
 configRxHBF2
 CX9261A, [20](#)
 configRxHBF3
 CX9261A, [21](#)
 configRxMixer
 CX9261A, [21](#)
 configRxMultiPointsQec
 CX9261A, [21](#)
 configRxMultiPointsQecImpl
 CX9261A, [21](#)
 configRxNco
 CX9261A, [22](#)
 configRxNcoFreq
 CX9261A, [22](#)
 configRxNcoMode
 CX9261A, [22](#)
 configRxNcoMS
 CX9261A, [22](#)
 configRxTIA
 CX9261A, [23](#)
 configTemperatureCompensation
 CX9261A, [23](#)
 configTxBandwidth

CX9261A, [23](#)
 configTxBandwidthImpl
 CX9261A, [23](#)
 configTxBIST
 CX9261A, [24](#)
 configTxChannel
 CX9261A, [24](#)
 configTxChannelFrequency
 CX9261A, [24](#)
 configTxCIC
 CX9261A, [24](#)
 configTxFilters
 CX9261A, [25](#)
 configTxFIR
 CX9261A, [25](#)
 configTxFIRCoef
 CX9261A, [25](#)
 configTxHBF1
 CX9261A, [25](#)
 configTxHBF2
 CX9261A, [26](#)
 configTxHBF3
 CX9261A, [26](#)
 configTxMixer
 CX9261A, [26](#)
 configTxMultiPointsQec
 CX9261A, [26](#)
 configTxNco
 CX9261A, [28](#)
 configTxNcoFreq
 CX9261A, [28](#)
 count
 CX9261A::AGCDIG, [39](#)
 CX9261A::AGCLMT, [41](#)
 CX9261A, [1](#)
 afcBBPll, [10](#)
 afcPll1, [10](#)
 afcPll2, [10](#)
 afcPll3, [11](#)
 afcPll4, [11](#)
 afcPll5, [11](#)
 Alert, [9](#)
 Band, [9](#)
 beforeQecState, [11](#)
 Bisec, [7](#)
 BIST, [8](#)
 Bypass, [8](#)
 Calib, [9](#)
 CalibMethod, [7](#)
 ceil, [11](#)
 ceil64, [12](#)
 clamp, [12](#)
 CMOS, [7](#)
 configADCCalib, [12](#)
 configAGC, [12](#)
 configAGCDIG, [12](#)
 configAGCLMT, [13](#)
 configAGCMode, [13](#)

configAGCPDT, 13
configBandwidthImpl, 13
configBBPLL, 13
configCalibMethod, 14
configDcCalib, 14
configDcCalibMethod, 14
configDcHighPassFactor, 14
configDcTrackingLength, 14
configFlashCalib, 15
configFPGA, 15
configInterface, 15
configPII1, 15
configPII2, 15
configPII3, 16
configPII4, 16
configPII5, 16
configPIIImpl, 17
configPIISource, 17
configPowerMode, 17
configPowerPins, 17
configQecCalib, 18
configRef, 18
configRefDoubler, 18
configRISCV, 18
configRxBandwidth, 18
configRxChannel, 19
configRxCIC, 19
configRxFilters, 19
configRxFIR, 20
configRxFIRCoef, 20
configRxHBF1, 20
configRxHBF2, 20
configRxHBF3, 21
configRxMixer, 21
configRxMultiPointsQec, 21
configRxMultiPointsQecImpl, 21
configRxNco, 22
configRxNcoFreq, 22
configRxNcoMode, 22
configRxNcoMS, 22
configRxTIA, 23
configTemperatureCompensation, 23
configTxBandwidth, 23
configTxBandwidthImpl, 23
configTxBIST, 24
configTxChannel, 24
configTxChannelFrequency, 24
configTxCIC, 24
configTxFilters, 25
configTxFIR, 25
configTxFIRCoef, 25
configTxHBF1, 25
configTxHBF2, 26
configTxHBF3, 26
configTxMixer, 26
configTxMultiPointsQec, 26
configTxNco, 28
configTxNcoFreq, 28
DAC, 9
DCCalibMethod, 7
delay, 38
Down, 8
Dual, 8
Duplex, 9
FDD, 10
floor, 28
floor64, 28
frac2bin, 28
HighPass, 7
initialBBPLL, 29
initialChip, 29
initialPLL1, 29
initialPLL2, 29
initialPLL3, 29
initialPLL4, 29
initialPLL5, 29
initialPowerConfig, 29
initialRxChannel, 30
initialTxChannel, 30
IOMode, 7
loadRxCalibPredis, 30
loadTxCalibPredis, 30
LoSource, 7
LVDS_E, 7
LVDS_S, 7
Master, 8
MultiPoints, 7
NCOMode, 8
Obs, 9
Off, 8
pause, 31
PII1, 8
PII2, 8
PII3, 8
PII4, 8
PII5, 8
PortMode, 8
PowerMode, 8
preparePIIForRxCalib, 31
preparePIIForTxCalib, 31
preparePowerForRxCalib, 32
preparePowerForTxCalib, 32
qecDelay, 38
R1T1, 10
R2T2, 10
readRxCalibPredis, 32
readSpi, 32
readTxCalibPredis, 33
Receive, 9
restoreQecState, 33
round, 33
round64, 33
Rx, 10
Rx1, 9
Rx2, 9
Rx3, 9

- rxCalib, 33
- RxChannel, 9
- rxDelayCalib, 34
- rxFFTCalib, 34
- rxTSTCalib, 34
- saveRxCalibPredis, 34
- saveTxCalibPredis, 35
- setPin, 35
- Shutdown, 9
- Single, 8
- Slaver, 8
- Sleep, 9
- TemperatureComp, 9
- toString, 35, 36
- Tracking, 7
- Transmit, 9
- TrMode, 9
- TST, 7
- Tx, 10
- Tx1, 10
- Tx2, 10
- txBisectionDcCalib, 36
- txBisectionQecCalib, 36
- txCalib, 36
- TxChannel, 10
- txTSTCalib, 37
- txTSTPhaseDelay, 37
- Up, 8
- waitRxTSTFinished, 37
- WorkMode, 10
- writeSpi, 37
- CX9261A::AGC, 38
 - dig, 38
 - lmt, 38
 - mode, 39
 - pdt, 39
- CX9261A::AGCDIG, 39
 - count, 39
 - threshold1, 39
 - threshold2, 40
 - threshold3, 40
 - threshold4, 40
- CX9261A::AGCLMT, 40
 - count, 41
 - threshold1, 41
 - threshold2, 41
 - threshold3, 41
 - threshold4, 41
- CX9261A::AGCPDT, 41
 - pdtM, 42
 - pdtN, 42
- CX9261A::Calib, 42
 - calibDC, 43
 - dcCalibMethod, 43
 - dcHighPassFactor, 43
 - dcTrackingLength, 43
 - f0, 43
 - rx1CalibPll, 43
 - rx2CalibPll, 43
 - rx3CalibPll, 43
 - rxMethod, 43
 - rxMultiPoints, 44
 - tx1CalibPll, 44
 - tx2CalibPll, 44
 - txMethod, 44
 - txMultiPoints, 44
- CX9261A::CalibPredis, 44
 - bHigh, 45
 - bLow, 45
 - cHigh, 45
 - cLow, 45
 - dcHigh, 45
 - dcLow, 45
 - dcqHigh, 45
 - dcqLow, 45
- CX9261A::Clock, 45
 - adc, 46
 - bbUseIntDiv, 46
 - ref, 46
 - refDiv, 46
 - refDoubler, 46
- CX9261A::CX9261AConfiguration, 47
 - agc, 48
 - calib, 48
 - clock, 48
 - interface, 48
 - pllFreq, 48
 - pllSource, 48
 - powerManagement, 48
 - rx1Config, 48
 - rx2Config, 49
 - rx3Config, 49
 - rxFilters, 49
 - rxFIRCoef, 49
 - rxNco, 49
 - tempComp, 49
 - tx1Config, 49
 - tx2Config, 49
 - txFilters, 50
 - txFIRCoef, 50
 - txNco, 50
- CX9261A::Filters, 50
 - cic, 51
 - fir, 51
 - hbf1, 51
 - hbf2, 51
 - hbf3, 51
- CX9261A::Interface, 51
 - ioMode, 52
 - portMode, 52
 - trMode, 52
 - workMode, 52
- CX9261A::PllFreq, 52
 - pll1Freq, 53
 - pll2Freq, 53
 - pll3Freq, 53

- pll4Freq, [53](#)
- pll5Freq, [53](#)
- rxExtFreq, [53](#)
- txExtFreq, [53](#)
- CX9261A::PllSource, [54](#)
 - rx1Source, [54](#)
 - rx2Source, [54](#)
 - rx3Source, [54](#)
 - rxUseExt, [54](#)
 - tx1Source, [55](#)
 - tx2Source, [55](#)
 - txUseExt, [55](#)
- CX9261A::Power, [55](#)
 - auxadc1, [56](#)
 - auxadc2, [56](#)
 - auxdac, [56](#)
 - bbpll, [56](#)
 - bgr, [56](#)
 - dig_rx1, [56](#)
 - dig_rx2, [56](#)
 - dig_rx3, [57](#)
 - dig_tx1, [57](#)
 - dig_tx2, [57](#)
 - exlo, [57](#)
 - fref1_rx, [57](#)
 - pll1, [57](#)
 - pll1_lldo, [57](#)
 - pll2, [57](#)
 - pll2_lldo, [57](#)
 - pll3, [57](#)
 - pll3_lldo, [58](#)
 - pll4, [58](#)
 - pll4_lldo, [58](#)
 - pll5, [58](#)
 - pll5_lldo, [58](#)
 - rx1, [58](#)
 - rx2, [58](#)
 - rx3, [58](#)
 - temp_sensor, [58](#)
 - tx1, [58](#)
 - tx2, [59](#)
- CX9261A::PowerManagement, [59](#)
 - fddPower, [59](#)
 - powerMode, [59](#)
 - rxPower, [59](#)
 - txPower, [59](#)
- CX9261A::Register, [60](#)
 - operator Type, [60](#)
 - operator[], [60](#)
 - Register, [60](#)
 - replace, [60](#)
 - value, [61](#)
- CX9261A::RxConfig, [61](#)
 - bandwidth, [61](#)
 - mixer, [61](#)
 - tia, [61](#)
- CX9261A::RxNCO, [62](#)
 - rxNCOBypass, [62](#)
 - rxNCOFreq, [62](#)
 - rxNCOMaster, [62](#)
 - rxNCOUpward, [62](#)
 - rxUseBist, [63](#)
- CX9261A::TxConfig, [63](#)
 - bandwidth, [63](#)
 - mixer, [63](#)
- CX9261A::TxNCO, [64](#)
 - txNCOFreq, [64](#)
 - txUseBist, [64](#)
- cx9261adriver.cpp, [64](#)
- cx9261adriver.h, [68](#)
 - LOGGING, [73](#)
 - VERSION, [73](#)
- CX9261A驱动, [1](#)
- DAC
 - CX9261A, [9](#)
- DCCalibMethod
 - CX9261A, [7](#)
- dcCalibMethod
 - CX9261A::Calib, [43](#)
- dcHighPassFactor
 - CX9261A::Calib, [43](#)
- dcHigh
 - CX9261A::CalibPredis, [45](#)
- dcLow
 - CX9261A::CalibPredis, [45](#)
- dcqHigh
 - CX9261A::CalibPredis, [45](#)
- dcqLow
 - CX9261A::CalibPredis, [45](#)
- dcTrackingLength
 - CX9261A::Calib, [43](#)
- defs.h, [74](#)
- delay
 - CX9261A, [38](#)
- dig
 - CX9261A::AGC, [38](#)
- dig_rx1
 - CX9261A::Power, [56](#)
- dig_rx2
 - CX9261A::Power, [56](#)
- dig_rx3
 - CX9261A::Power, [57](#)
- dig_tx1
 - CX9261A::Power, [57](#)
- dig_tx2
 - CX9261A::Power, [57](#)
- Down
 - CX9261A, [8](#)
- Dual
 - CX9261A, [8](#)
- Duplex
 - CX9261A, [9](#)
- exlo
 - CX9261A::Power, [57](#)

- f0
 - CX9261A::Calib, [43](#)
- FDD
 - CX9261A, [10](#)
- fddPower
 - CX9261A::PowerManagement, [59](#)
- fir
 - CX9261A::Filters, [51](#)
- floor
 - CX9261A, [28](#)
- floor64
 - CX9261A, [28](#)
- frac2bin
 - CX9261A, [28](#)
- fref1_rx
 - CX9261A::Power, [57](#)
- hbf1
 - CX9261A::Filters, [51](#)
- hbf2
 - CX9261A::Filters, [51](#)
- hbf3
 - CX9261A::Filters, [51](#)
- HighPass
 - CX9261A, [7](#)
- initialBBPLL
 - CX9261A, [29](#)
- initialChip
 - CX9261A, [29](#)
- initialPLL1
 - CX9261A, [29](#)
- initialPLL2
 - CX9261A, [29](#)
- initialPLL3
 - CX9261A, [29](#)
- initialPLL4
 - CX9261A, [29](#)
- initialPLL5
 - CX9261A, [29](#)
- initialPowerConfig
 - CX9261A, [29](#)
- initialRxChannel
 - CX9261A, [30](#)
- initialTxChannel
 - CX9261A, [30](#)
- interface
 - CX9261A::CX9261AConfiguration, [48](#)
- IOMode
 - CX9261A, [7](#)
- ioMode
 - CX9261A::Interface, [52](#)
- Imt
 - CX9261A::AGC, [38](#)
- loadRxCalibPredis
 - CX9261A, [30](#)
- loadTxCalibPredis
 - CX9261A, [30](#)
- LOGGING
 - cx9261adriver.h, [73](#)
- LoSource
 - CX9261A, [7](#)
- LVDS_E
 - CX9261A, [7](#)
- LVDS_S
 - CX9261A, [7](#)
- Master
 - CX9261A, [8](#)
- mixer
 - CX9261A::RxConfig, [61](#)
 - CX9261A::TxConfig, [63](#)
- mode
 - CX9261A::AGC, [39](#)
- MultiPoints
 - CX9261A, [7](#)
- NCOMode
 - CX9261A, [8](#)
- Obs
 - CX9261A, [9](#)
- Off
 - CX9261A, [8](#)
- operator Type
 - CX9261A::Register, [60](#)
- operator[]
 - CX9261A::Register, [60](#)
- pause
 - CX9261A, [31](#)
- pdt
 - CX9261A::AGC, [39](#)
- pdtM
 - CX9261A::AGCPDT, [42](#)
- pdtN
 - CX9261A::AGCPDT, [42](#)
- PII1
 - CX9261A, [8](#)
- pll1
 - CX9261A::Power, [57](#)
- pll1_Ido
 - CX9261A::Power, [57](#)
- pll1Freq
 - CX9261A::PIIFreq, [53](#)
- PII2
 - CX9261A, [8](#)
- pll2
 - CX9261A::Power, [57](#)
- pll2_Ido
 - CX9261A::Power, [57](#)
- pll2Freq
 - CX9261A::PIIFreq, [53](#)
- PII3
 - CX9261A, [8](#)
- pll3
 - CX9261A::Power, [57](#)

- pll3_lldo
 - CX9261A::Power, 58
- pll3Freq
 - CX9261A::PllFreq, 53
- Pll4
 - CX9261A, 8
- pll4
 - CX9261A::Power, 58
- pll4_lldo
 - CX9261A::Power, 58
- pll4Freq
 - CX9261A::PllFreq, 53
- Pll5
 - CX9261A, 8
- pll5
 - CX9261A::Power, 58
- pll5_lldo
 - CX9261A::Power, 58
- pll5Freq
 - CX9261A::PllFreq, 53
- pllFreq
 - CX9261A::CX9261AConfiguration, 48
- pllSource
 - CX9261A::CX9261AConfiguration, 48
- PortMode
 - CX9261A, 8
- portMode
 - CX9261A::Interface, 52
- powerManagement
 - CX9261A::CX9261AConfiguration, 48
- PowerMode
 - CX9261A, 8
- powerMode
 - CX9261A::PowerManagement, 59
- preparePllForRxCalib
 - CX9261A, 31
- preparePllForTxCalib
 - CX9261A, 31
- preparePowerForRxCalib
 - CX9261A, 32
- preparePowerForTxCalib
 - CX9261A, 32
- qecDelay
 - CX9261A, 38
- R1T1
 - CX9261A, 10
- R2T2
 - CX9261A, 10
- readme.txt, 80
- readRxCalibPredis
 - CX9261A, 32
- readSpi
 - CX9261A, 32
- readTxCalibPredis
 - CX9261A, 33
- Receive
 - CX9261A, 9
- ref
 - CX9261A::Clock, 46
- refDiv
 - CX9261A::Clock, 46
- refDoubler
 - CX9261A::Clock, 46
- Register
 - CX9261A::Register, 60
- replace
 - CX9261A::Register, 60
- restoreQecState
 - CX9261A, 33
- round
 - CX9261A, 33
- round64
 - CX9261A, 33
- Rx
 - CX9261A, 10
- Rx1
 - CX9261A, 9
- rx1
 - CX9261A::Power, 58
- rx1CalibPll
 - CX9261A::Calib, 43
- rx1Config
 - CX9261A::CX9261AConfiguration, 48
- rx1Source
 - CX9261A::PllSource, 54
- Rx2
 - CX9261A, 9
- rx2
 - CX9261A::Power, 58
- rx2CalibPll
 - CX9261A::Calib, 43
- rx2Config
 - CX9261A::CX9261AConfiguration, 49
- rx2Source
 - CX9261A::PllSource, 54
- Rx3
 - CX9261A, 9
- rx3
 - CX9261A::Power, 58
- rx3CalibPll
 - CX9261A::Calib, 43
- rx3Config
 - CX9261A::CX9261AConfiguration, 49
- rx3Source
 - CX9261A::PllSource, 54
- rxCalib
 - CX9261A, 33
- RxChannel
 - CX9261A, 9
- rxDelayCalib
 - CX9261A, 34
- rxExtFreq
 - CX9261A::PllFreq, 53
- rxFFTCalib
 - CX9261A, 34

- rxFilters
 - CX9261A::CX9261AConfiguration, 49
- rxFIRCoef
 - CX9261A::CX9261AConfiguration, 49
- rxMethod
 - CX9261A::Calib, 43
- rxMultiPoints
 - CX9261A::Calib, 44
- rxNco
 - CX9261A::CX9261AConfiguration, 49
- rxNCOBypass
 - CX9261A::RxNCO, 62
- rxNCOFreq
 - CX9261A::RxNCO, 62
- rxNCOMaster
 - CX9261A::RxNCO, 62
- rxNCOUpward
 - CX9261A::RxNCO, 62
- rxPower
 - CX9261A::PowerManagement, 59
- rxTSTCalib
 - CX9261A, 34
- rxUseBist
 - CX9261A::RxNCO, 63
- rxUseExt
 - CX9261A::PllSource, 54
- saveRxCalibPredis
 - CX9261A, 34
- saveTxCalibPredis
 - CX9261A, 35
- setPin
 - CX9261A, 35
- Shutdown
 - CX9261A, 9
- Single
 - CX9261A, 8
- Slaver
 - CX9261A, 8
- Sleep
 - CX9261A, 9
- temp_sensor
 - CX9261A::Power, 58
- tempComp
 - CX9261A::CX9261AConfiguration, 49
- TemperatureComp
 - CX9261A, 9
- threshold1
 - CX9261A::AGCDIG, 39
 - CX9261A::AGCLMT, 41
- threshold2
 - CX9261A::AGCDIG, 40
 - CX9261A::AGCLMT, 41
- threshold3
 - CX9261A::AGCDIG, 40
 - CX9261A::AGCLMT, 41
- threshold4
 - CX9261A::AGCDIG, 40
- CX9261A::AGCLMT, 41
- tia
 - CX9261A::RxConfig, 61
- toString
 - CX9261A, 35, 36
- Tracking
 - CX9261A, 7
- Transmit
 - CX9261A, 9
- TrMode
 - CX9261A, 9
- trMode
 - CX9261A::Interface, 52
- TST
 - CX9261A, 7
- Tx
 - CX9261A, 10
- Tx1
 - CX9261A, 10
- tx1
 - CX9261A::Power, 58
- tx1CalibPll
 - CX9261A::Calib, 44
- tx1Config
 - CX9261A::CX9261AConfiguration, 49
- tx1Source
 - CX9261A::PllSource, 55
- Tx2
 - CX9261A, 10
- tx2
 - CX9261A::Power, 59
- tx2CalibPll
 - CX9261A::Calib, 44
- tx2Config
 - CX9261A::CX9261AConfiguration, 49
- tx2Source
 - CX9261A::PllSource, 55
- txBisectionDcCalib
 - CX9261A, 36
- txBisectionQecCalib
 - CX9261A, 36
- txCalib
 - CX9261A, 36
- TxChannel
 - CX9261A, 10
- txExtFreq
 - CX9261A::PllFreq, 53
- txFilters
 - CX9261A::CX9261AConfiguration, 50
- txFIRCoef
 - CX9261A::CX9261AConfiguration, 50
- txMethod
 - CX9261A::Calib, 44
- txMultiPoints
 - CX9261A::Calib, 44
- txNco
 - CX9261A::CX9261AConfiguration, 50
- txNCOFreq

- CX9261A::TxNCO, [64](#)
- txPower
 - CX9261A::PowerManagement, [59](#)
- txTSTCalib
 - CX9261A, [37](#)
- txTSTPhaseDelay
 - CX9261A, [37](#)
- txUseBist
 - CX9261A::TxNCO, [64](#)
- txUseExt
 - CX9261A::PllSource, [55](#)
- Up
 - CX9261A, [8](#)
- utils.h, [80](#)
- value
 - CX9261A::Register, [61](#)
- VERSION
 - cx9261adriver.h, [73](#)
- waitRxTSTFinished
 - CX9261A, [37](#)
- WorkMode
 - CX9261A, [10](#)
- workMode
 - CX9261A::Interface, [52](#)
- writeSpi
 - CX9261A, [37](#)