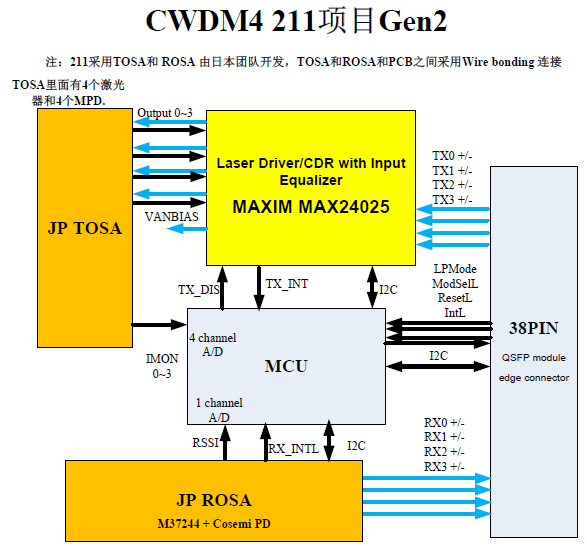
QSFP28光模块软件设计

1. **QSFP28光模块简介**

该项目主要为了满足市场对高速光模块的需求而研发，主要应用于企业的数据中心，这个项目主要针对CWDM4中距离传输的应用场景。100G QSFP28光模块采用的是SLIABS EFM系列MCU+ MAXIM24025 TX LASER DRIVER + MACOM RX 37244 + MACOM TOSA/ROSA，其中TX的4 CHANNELS一一对齐，包括顺序和P/N都对齐，RX的4个CHANNELS顺序反序(1对应4,2对应3， 3对应2， 4对应1)，每对差分线P/N反序。

**1.1 硬件结构图**



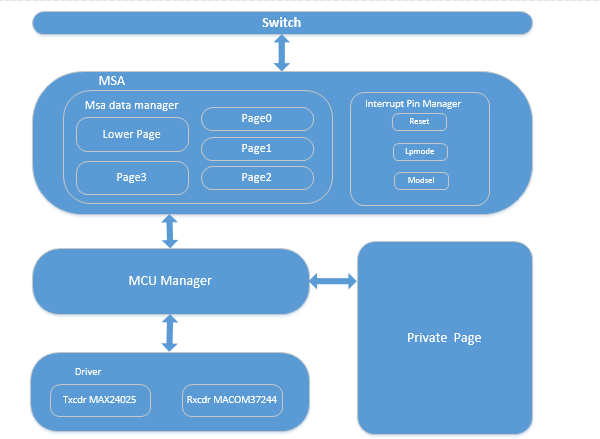
**1.2 MCU IO PORT定义**



1. **QSFP28光模块软件设计**

因为光模块的种类和产品迭代较快，所以需要考虑后续移植的问题，软件的架构设计尤为重要，以便后续快速的开发新的产品。另一方面，由于硬件资源有限(4K RAM + 64K FLASH)，所以软件的实现上要尽量短小精悍。软件开发主要分四大功能块: 一是MCU外设功能的实现，二是MSA协议标准的实现，三是对DRIVER控制的实现，四是MCU任务调度、产测命令以及调试方案的实现。

**2.1 软件结构图**

****

**2.2 MCU软件流程图**



**2.3 软件数据结构表**

* MSA LOWER对应MSA LOWER PAGE。
* P0对应MSA PAGE0。
* P1对应MSA PAGE1。
* P2对应MSA PAGE2。
* P3对应MSA PAGE3。
* P4-MCU是私有PAGE4，整个PAGE的属性都是RAM，主要保存DDMI的原始数据和DEBUG命令的相关数据。
* P5-HW\_INFO私有PAGE5，整个PAGE的属性都是FALSH，主要保存老化信息和模块类型，硬件版本，PCBA\_CODE, LASER\_TX\_SN, LASER\_RX\_SN, PCBA\_SN等。
* P6-LUT\_1, P7-LUT\_2, P8-LUT\_3是私有PAGE6，PAGE7，PAGE8，3个PAGE的属性都是FALSH，基本上都是保存校准参数，其中PAGE6和PAGE7主要保存LUT TABLE的参数以及DDMI校准的参数， PAGE8为其它的校准参数，如AVG RX\_LOS的校准参数等。
* P9-AGE是私有PAGE9，整个PAGE的属性都是FALSH，主要保存老化结果信息。
* P10-INIT是私有PAGE10，整个PAGE的属性都是FALSH，主要用于保存LASER DRIVER寄存器的配置值，暂时没有使用到。

相关的具体信息见下表:

****

**2.4 MCU外设功能的实现**

外设功能模块主要是MCU外设驱动的开发，包括ADC，DAC，TIMER，I2C SLAVE，I2C MASTER，EXTERN INTERRUPT，PORT MATCH等。

**2.4.1 ADC功能**

因为ADC的参考电压的稳定决定着ADC数值的准确性，所以CWDM4-211项目的ADC选取的是MCU内部1.65V的参考电压，因为它比较稳定，然后将ADC的PGA GAIN设置成0.5，那么ADC的采样范围就变成了0~3.3V。CWDM4-211项目的ADC主要应用于DDMI数据的采样，包括TEMP，VCC，RSSI，TXMON。TEMP采用MCU内部自带的温度传感器来获取温度值，VCC，RSSI，TXMON通过采样MCU的IO PORT的电压来获取值。需要注意的是，RSSI是一个IO PORT对应4个CHANNELS，它是通过配置通道选择寄存器来切换通道，所以不能通过自动ADC POLL来采样，必须通过手动ADC采样，即切换通道寄存器以后，手动计算这个IO PORT的ADC值。

**2.4.2 DAC功能**

只有带有TEC功能的模块才会使用到这个功能，LR4-301的项目就需要支持DAC功能，CWDM4-211项目不需要。

**2.4.3 TIMER功能**

该MCU支持5个TIMER，其中TIMER0用于I2C MASTER的时钟源，TIMER 1用于UART时钟源，TIMER 2用于软件定时器的设计，TIMER 3和TIMER 4未使用，TIMER 5用于实现微秒级延时。考虑到有些任务需要周期性执行，并且不同的任务要求不一样的周期，所以通过TIMER2这个硬件定时器设计出了一套软件定时器，可以扩展出无数个不同周期的定期任务，极大的方面了后续定时任务的实现。

**2.4.4 I2C SLAVE功能**

I2C SLAVE功能是支持模块与HOST通信的，是HOST读写模块信息的唯一途径，该MCU的这个功能只有P2.0 PIN和P2.1 PIN支持。

**2.4.5 I2C MASTER功能**

I2C MASTER功能是模块内部通信的基石，它负责与DRIVER通信，用于复位，配置DRIVER和读取DRIVER的状态信息，该MCU支持多个IO可配置I2C MASTWR功能，CWDM4-211项目采用的是P1.2 PIN和P1.3 PIN。I2C MASTER需要注意SCL线被从设备恒拉低的情况，因为模块内部会使用I2C MASTER周期性访问CDR，那么以下两种情况会将SCL拉低:

* 主机向从机写数据或地址时，从机如果发出ACK应答，则会第9个CLK的期间拉低SDA
* 主机读数据的时候，从机会在BIT为0时对应的CLK期间拉低SDA

如果在CDR将SCL拉低的过程中，HOST主动将模块重启了，那么再次上电的过程中，就需要加入I2C MASTER复位的操作，确保总线被释放。

**2.4.6** EXTERN INTERRUPT**功能**

该MCU只支持EXTERN INTERRUPT0和EXTERN INTERRUPT1，其中EXTERN INTERRUPT0用于LPMODE的实现，EXTERN INTERRUPT1用于RESET的实现。

**2.4.7 PORT MATCH功能**

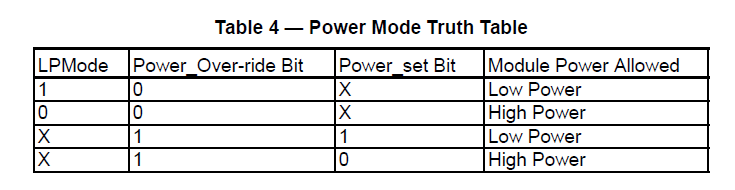
PORT MATCH也是一种中断机制并且是电平触发，唯一不同的是PORT MATCH机制是实际电平和目标电平不一致的时候会触发进入中断，这个功能是为了满足EXTERN INTERRUPT有限而设计的，如此一来，每个IO PORT都可以支持中断响应，CWDM4-211项目的TX INTERRUPT，RX INTERRUPT和MODSEL都是通过PORT MATCH机制实现的。

**2.5 MSA协议标准的实现**

**2.5.1 低速逻辑信号功能**

**(1) LPMODE功能**

模块上电以后，在未响应外部LPMODE的情况下默认处于低功耗模式，目的是防止上电过程中冲击电流过大，模块启动成功以后根据外部LPMODE和MSA LOWER PAGE 93地址的低2BITS的值来配置功耗模式，其中它们的对应关系如下表:



因为LPMODE是电平触发，并且默认是低电平触发，所以当外部LPMODE = 0时，会响应中断并切换到高功耗模式，当外部LPMODE = 1时，不会响应进入中断，那么模块保持默认状态，低功耗模式，与LPMODE当前的电平状态保持一致，但是这种处理方式依赖于默认值，不利于后续的代码的扩展和维护，所以在模块启动的过程中，增加根据LPMODE的电平状态来更新功耗模式的实现。

低功耗模式的实现方式是关闭TX发光和关闭RX OUTPUT(MSA PGAE3 BYTE241)，相反高功耗的实现方式是打开TX发光和打开RX OUTPUT(MSA PGAE3 BYTE241)。

**(2) MODSEL功能**

MODSEL采用的是PORT MATCH中断，本质也是电平触发，唯一不同的是PORT MATCH机制是电平不一致的情况下才触发进入中断，匹配电平默认为低，所以当外部MODSEL为低时，不触发进入中断，模块保持默认状态，I2C SLAVE功能默认是打开的，所以MODSEL电平状态和I2C SLAVE功能是保持一致的；当外部MODSEL为高时，与匹配电平不一致，触发进入中断，关闭I2C SLAVE功能，但是这种处理方式依赖于默认值，不利于后续的代码的扩展和维护，所以在模块启动的过程中，加入对MODSEL电平判断的逻辑，然后根据MODSEL的电平状态来开关I2C SLAVE。

MODSEL的实现方式是通过使能I2C SLAVE外设功能来实现的。

**(3)RESET功能**

**RESET**的实现方式为下降沿触发进入中断，但不执行RESET动作，进入中断后，关闭激光器发光，将INTL拉高，表示清除告警，然后一直等待RESET被释放到高电平才开始RESET模块。模块上电的时候需要检测RESET的电平状态，如果RESET= 0，那么模块被挂住，不能正常启动，不响应外部的任何请求。RESET的实现方式是通过配置MCU的软重启寄存器来实现的，因为MCU重启以后会重新复位和初始化TXCDR、RXCDR，但是需要注意的是重新复位和初始化CDR并不是最先执行的， RESET后首先配置MCU的外设， 然后初始化MCU RAM， 最后才是复位和初始化CDR，这个过程的时间较长，那么从RESET开始到复位和初始化CDR之前，CDR都是保持重启之前的状态，如果重启之前是发光状态，那么这个过程中，也是出于发光状态，这显然是不合理的，所以需要在进入复位状态并且没有复位之前，要将TXCDR和RXCDR恢复到默认状态。

**(4)MODPRS功能**

这个PIN直接由硬件控制，软件不控制，模块上此PIN接地，主板上拉高, 主板通过检测此PIN电平确认模块是否接入。

**(5)INTL 功能**

**INTL**当且仅当在模块发生异常的时候将INTL拉低表示告警，告诉HOST有异常并来读取异常状态。

INTL的实现逻辑包括以下几点:

* 模块上电以后到READY状态，主动将INTL拉低并一直保持，即便这期间没有任何的中断告警，也一直保持低电平直到HOST通过I2C来读取一次状态，读取这次状态结束以后，再判断是否有中断告警，如果没有，将INTL拉高，如果有，继续保持低电平；
* 模块的高低功耗切换，不上报TX和RX的所有告警，所以INTL的电平变化不受功耗变化的影响。如果在高功耗模式下就有TX、RX的相关告警，进入低功耗模式以后，这些告警依然保留直到HOST来读取一遍，当HOST读取一遍清除这次告警以后，不再产生TX、RX的任何告警，当恢复到高功耗模式时，如果有TX、RX告警，再次产生相应的告警。
* INTL的电平变化受两个状态控制:

1. 模块产生中断告警；
2. 对应的中断告警的掩码功能打开；

**2.5.2 安全等级功能**

MSA协议要求LOWER PAGE BYTE123 ~ BYTE126是存放密码相关的，CWDM4-211的模块的默认密码是“0x8F 0x9E 0xAD 0xEC”，只有进入的安全等级以后，才能访问私有的PAGE页，否则只能访问MSA 协议标准PAGE，同时安全等级也控制着MSA标准PAGE的读写和是否掉电保存的权限。

**2.5.3 MSA标准页的默认值和访问权限功能**

具体的每个BYTE的默认值和访问权限见下表:



**2.5.4 MSA DDMI 功能**

(1)TEMP使用的是MCU自带的温度传感器，通过ADC采样转换得到的。因为MCU温度传感器得到的是MCU芯片的温度，模块校准后的温度表示模块外壳的温度，所以TEMP的校准是通过减去一个正数值OFFSET来实现的。当在负温度环境下，例如-5摄氏度的环境下，如果上电后直接减去这个正数值OFFSET，那么上报的温度就是比-5还要低的温度，反馈给人的现象就是模块有“自动降温”的功能，这显然不合理，所以引入了温度慢启动机制，即在上电启动过程中，通过公式OFFSET \* (t / 180)来实现，其中t是启动的时间，单位为秒，从上电后每秒加1，所以到达180秒超时之后，公式计算的结果就是实际的OFFSET。温度慢启动虽然解决了负温度环境下的“自动降温”的问题，但是它同样会引入新的问题，例如因为模块每次上电或者软重启以后，公式OFFSET \* (t / 180)就会重新计算一次，那么当模块处于正温度环境下并且温度稳定以后，每次软重启温度需要180秒才能稳定，这显然也是不满足实际要求的。为了处理这个问题，每次重启以后，读取MCU RSTSRC 寄存器，判断MCU的重启类型，如果是软重启，就关闭温度慢启动，否则就是上电重启，开启温度慢启动。

(2)VCC是通过ADC采样外部PIN的电压转换得到的, VCC的校准也是通过加减一个OFFSET来实现的。

(3)RSSI是通过ADC采样外部PIN的电压转换得到的。需要注意的是，RSSI是一个PIN对应4个CHANNEL，通过通道寄存器来切换通道，因此不能通过自动ADC POLL来采样，必须通过手动ADC采样，即切换通道寄存器以后，手动计算这个IO PORT的ADC值。RSSI的校准是通过选取两个点RSSI1和RSSI2，然后分别测量出它们对应的POWER(DB)值，通过这两个点来拟合直线。当TX关闭发光或者进入低功耗模式的时候，RXPOWER上报的值为-40(DBM)。

(4)TXMON是通过ADC采样外部PIN的电压转换得到的, TXMON的校准是通过选取两个点TXMON1和TXMON2，然后分别计算出它们对应的POWER(DB)值，通过这两个点来拟合直线。当TX关闭发光或者进入低功耗模式的时候，TXPOWER的结果为-40(DBM)。

(5)BIAS是通过直接读取TXCDR MAX24025的寄存器后做一定的计算得到的。当TX关闭发光或者进入低功耗模式的时候，BIAS的结果为0(mA)。

需要注意的是，由于DDMI DATA都是双字节宽度，并且它们是实时更新的，那么就有可能出现只更新其中一个字节的时候，HOST来读取数据了，这样的数据肯定是错误的。CWDM4-211项目采取的解决方案是上锁，在更新数据之前将当前数据备份一份，然后上锁，更新数据，更新完数据以后解锁。如果在已经上锁并且还未解锁的情况下HOST来读取数据，因为检测已经上锁了，那么返回备份中的数据，并记录当前读的是备份数据，当HOST来读取第二个BYTE的时候， 因为已经记录了上次读的是备份数据，所以这次读取依然返回备份中的数据。需要注意这里最好不要通过关闭I2C SLAVE来实现，这样很容易导致HOST的访问被拒绝。

**2.6 DRIVER的控制**

DRIVER的控制主要包括对其复位，初始化和APC(AUTO POWER CONTROL)。APC功能通过AUTO [ENABLE|DISABLE]命令开关并且断电失效，默认状态是AUTO ENABLE。因为APC的唯一变量是温度，所以只有当温度变化的时候，APC才会工作，为了降低MCU的无效工作的处理，当温度变化的幅度在1摄氏度以内的时候，通过APC算法计算出的BIAS，MOD和CROSSING电流基本上是不变的，所以这个时候APC是不工作的。

**2.7软件任务调度算法、生产校准命令和调试方案的实现**

主程序运行多个POLL任务，其中有些任务是跟中断相关的，所以当中断发生以后，需要将其相关的任务优先调度，以满足协议的时序要求。由于CWDM4-211项目是裸机程序，没有OS，所以任务调度必须自己实现。实现的方案如下: 给每个POLL任务分配一个ID(注意这里不能直接使用函数名代替ID，以为函数名就是函数指针，KEIL编译器不支持函数的参数为指针)，当中断发生时，在中断处理函数中，将这个ID记录起来，然后调度任务在每个POLL任务结束后，先判断是否有快速调度发生，如果有，优先执行快速调用，等待快速调用任务执行结束后，继续按照之前的顺序执行下一个POLL任务。需要特别注意SIMPLICITY STUDIO的KEIL编译器为了降低编译后版本的大小，不支持直接使用函数指针，凡是使用函数指针，必须手动设置编译过程，调整正确的调用顺序，具体的实现方法参考工程项目中的OVERLAY.LINK文件。

为了使模块可以正常的进行生产校准，有些生产校准命令是必不可少的，如EEP INIT，FACTORY\_RESET，AUTO [ENABLE|DISABLE]命令的响应等， 再例如要求可以直接访问DRIVER，获取异常状态，同时也实现了记录LOG的机制，HOST可以通过读取模块的LOG来验证软件的逻辑是否正确，软件执行的流程按照如期等问题。LOG机制复用MSA PAGE2，通过宏编译决定是否使用，并且只用于DEBUG调试使用，并不影响模块其他任何功能，也不占用任何的软硬件资源。