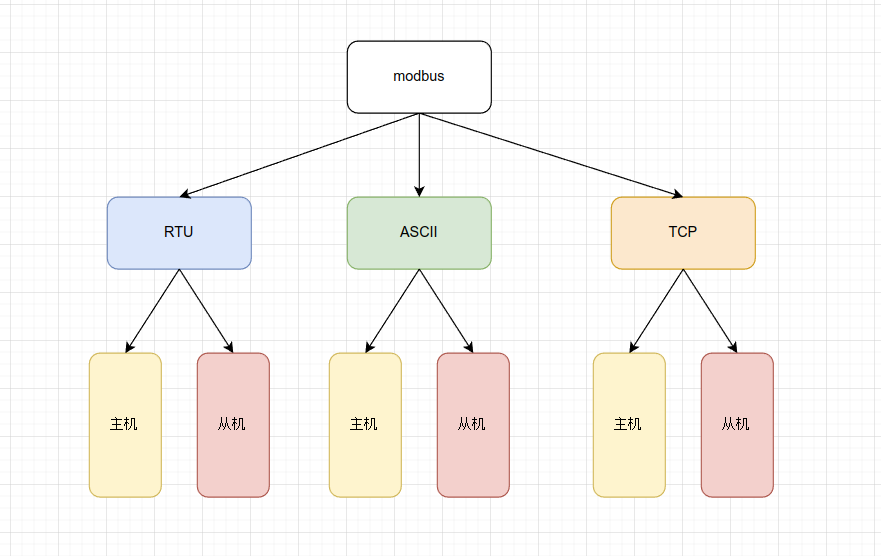
 modbus\_module 设计概要

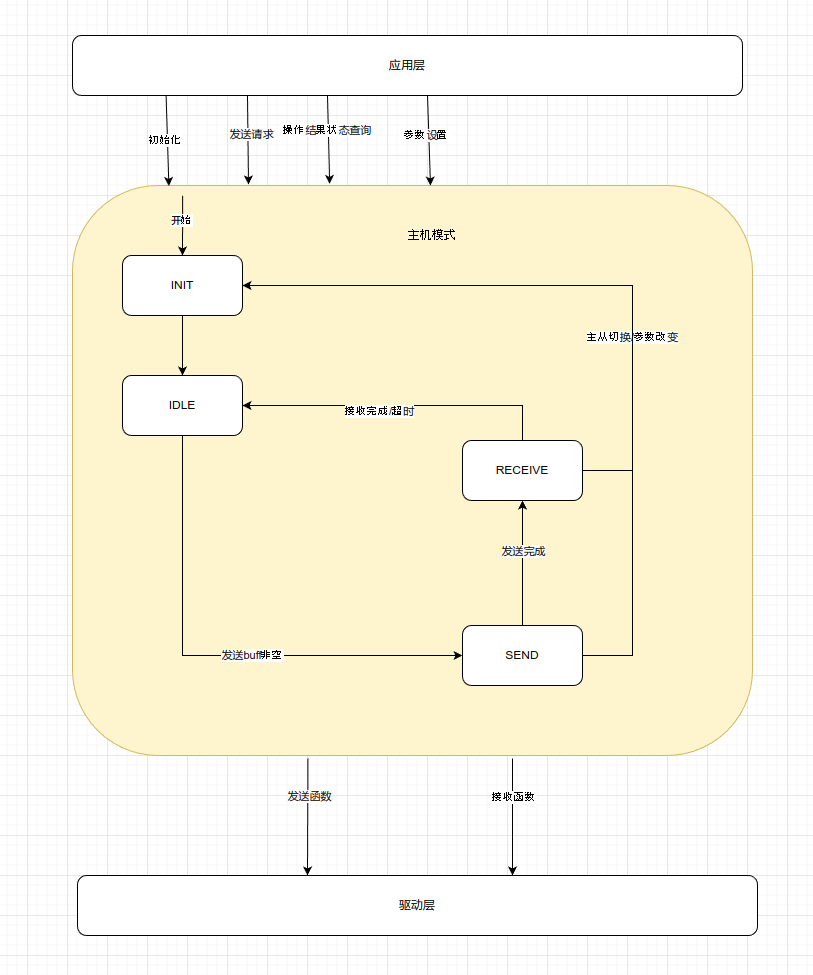
1. 代码结构
   1. 源码结构
   2. 主机模式
   3. 从机模式
2. 数据类型
3. 接口设计
4. 移植/应用
   1. 回调接口
   2. 移植接口

**modbus\_module** 设计概要

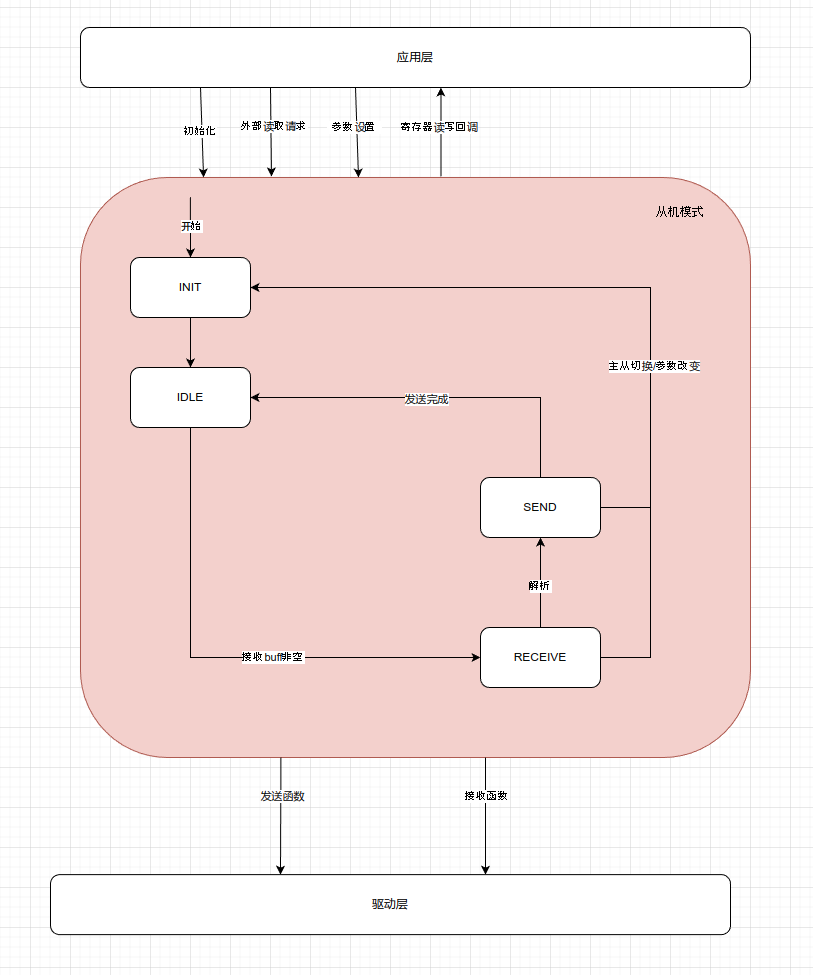
1. 代码结构
   1. 源码结构



* 1. 主机模式



* 1. 从机模式



typedef struct Modebus\_RTU\_Fun\_Table

{

uint8\_t fcode;

int8\_t (\*request)(stModbus\_RTU\_Handler \*handler, stModbus\_RTU\_Sender \*sender); int8\_t (\*slave\_parse)(stModbus\_RTU\_Handler \*handler, uint8\_t \*buff, uint16\_t

len);

1. 数据类型

int8\_t (\*master\_parse)(stModbus\_RTU\_Handler \*handler, uint8\_t \*buff, uint16\_t len);

}stModebus\_RTU\_Fun\_Table;

typedef struct Modbus\_RTU\_Handler

{

uint8\_t dev\_addr;

stModbus\_RTU\_State state, last\_state; // send/receive switch judge

emModebus\_RTU\_Mode mode, last\_mode; // slave/master switch judge

uint32\_t last\_call\_tick; uint8\_t tx\_buff[300];

uint16\_t tx\_len;

uint8\_t rx\_buff[300]; uint16\_t rx\_len;

uint16\_t \*master\_parse\_addr; // master parse address(master mode use only)

uint32\_t Master\_Wait\_Count; // master wait recv time count(master mode use only)

uint32\_t Master\_Wait\_Recv\_Limt; // master wait recv limit(master mode use only)

stModebus\_RTU\_Fun\_Table \*fun\_table; // callback function table

uint8\_t fun\_table\_items;

int8\_t (\*send)(uint8\_t \*buff, uint16\_t len); int8\_t (\*recv)(uint8\_t \*buff, uint16\_t \*len);

int8\_t reg\_map\_id; // reserve for

select register map table

int8\_t (\*read\_input)(stModbus\_RTU\_InputReader \*reader); int8\_t (\*read\_hold)(stModbus\_RTU\_HoldReader \*reader);

int8\_t (\*write\_hold)(stModbus\_RTU\_HoldWriter \*writer);

}stModbus\_RTU\_Handler;

typedef struct Modbus\_RTU\_Handler\_Attribute

{

uint8\_t dev\_addr;

emModebus\_RTU\_Mode mode;

int8\_t (\*send)(uint8\_t \*buff, uint16\_t len); int8\_t (\*recv)(uint8\_t \*buff, uint16\_t \*len);

int8\_t reg\_map\_id; // reserve for

select register map table

int8\_t (\*read\_input)(stModbus\_RTU\_InputReader \*reader); int8\_t (\*read\_hold)(stModbus\_RTU\_HoldReader \*reader);

int8\_t (\*write\_hold)(stModbus\_RTU\_HoldWriter \*writer);

stModebus\_RTU\_Fun\_Table \*fun\_table; // callback function table

uint8\_t fun\_table\_items;

}stModbus\_RTU\_Handler\_Attr;

1. 接口设计

int8\_t (\*send)(uint8\_t \*buff, uint16\_t len);

int8\_t (\*recv)(uint8\_t \*buff, uint16\_t \*len); int8\_t reg\_map\_id;

// 驱动层读接口

// 驱动层写接口

// reserve for select

/\*\*

* @brief modbus rtu init
* @param handler: modbus rtu handler
* @param bus: modbus rtu bus
* @param attr: modbus rtu handler attr
* @return 0: success, -1: fail

\*/

int8\_t modbus\_rtu\_init(stModbus\_RTU\_Handler \*handler, emModebus\_RTU\_Bus bus, stModbus\_RTU\_Handler\_Attr \*attr);

void modbus\_rtu\_run(stModbus\_RTU\_Handler \*handler);

/\*\*

* @brief 读取寄存器
* @param bus 总线编号
* @param dev\_addr 设备地址
* @param reg\_addr 寄存器地址
* @param reg\_num 寄存器数量
* @param reg\_data 寄存器数据地址
* @return 0 成功，-1 失败

\*/

int8\_t modbus\_rtu\_read\_hold(emModebus\_RTU\_Bus bus, uint8\_t dev\_addr, uint16\_t reg\_addr, uint16\_t reg\_num, uint16\_t \*reg\_data);

/\*\*

* @brief 读取上次操作完成状态
* @param bus 总线编号
* @return 0 完成，1 未完成 -１失败

\*/

int8\_t modbus\_rtu\_opt\_status(emModebus\_RTU\_Bus bus);

1. 移植**/**应用
   1. 回调接口

register map table

int8\_t (\*read\_input)(stModbus\_RTU\_InputReader \*reader);

器回调

int8\_t (\*read\_hold)(stModbus\_RTU\_HoldReader \*reader);

器回调

int8\_t (\*write\_hold)(stModbus\_RTU\_HoldWriter \*writer);

器回调

// 应用层读输入寄存

// 应用层读保持寄存

// 应用层写保持寄存

* 1. 移植接口

uint32\_t modbus\_port\_get\_time\_ms();