// hand shake

Soc\_protocol.h里面：

#if 0

CHECK\_IN \【1】

#define CHECK\_IN \【1】

do { \

ConsoleU::writeLine(String::Format("{0} in", \_\_FUNCTION\_\_), ConsoleU::Level::Normal); \

} while (0)

CHECK\_OUT \【2】

#define CHECK\_OUT \【2】

do { \

ConsoleU::writeLine(String::Format("{0} out", \_\_FUNCTION\_\_), ConsoleU::Level::Normal); \

} while (0)

#else

#define CHECK\_IN

#define CHECK\_OUT

#endif

REPORT\_CMD(\_X\_) \【3】

#define REPORT\_CMD(\_X\_) \【3】

do { \

ConsoleU::writeLine(String::Format("process cmd({0})", \_X\_), ConsoleU::Level::Info); \

} while (0)

Soc\_protocol.cpp里面：

UartProtocol::pack【4】

Int32 UartProtocol::pack(Command cmd, array<Byte>^ data, array<Byte>^ \*pPacket) { 【4】

Int32 errCode = 0;

UInt16 lenTotal = CONTAINER\_LENGTH;

if (data != nullptr) {

lenTotal += data->Length;

}

array<Byte>^ packet = gcnew array<Byte>(lenTotal);

packet[(Int32) Index::startCode] = BYTE\_START\_CODE;【4-1】

packet[(Int32) Index::protocolType] = BYTE\_PROTOCOL\_TYPE;

packet[(Int32) Index::commandDirection] = (Byte) Direction::pcSend; 【4-2】

packet[(Int32) Index::commandCode] = (Byte) cmd;【4-3】

array<Byte>^ aLengTotal;

errCode = Packet::uIntToByteArray(lenTotal, &aLengTotal);【5】

if (errCode < 0) {

return errCode;

}

aLengTotal->CopyTo(packet, (Int32) Index::commnadLengthH);

// pack data

if (data != nullptr) {

data->CopyTo(packet, (Int32) Index::data);

}

Byte sum = 0;

for each (Byte cell in packet) {

sum += cell;

}

packet[packet->Length -1] = sum;

\*pPacket = packet;

return 0;

}

Soc\_protocol.h里面：

Index {【4-1】

public enum class Index {【4-1】

startCode,

protocolType,

commandDirection,

commandCode,

commnadLengthH,

commandLengthL,

data,

};

Direction {【4-2】

public enum class Direction {【4-2】

stbSend = 0xDA,

pcSend = 0xDB,

};

在soc\_protocol.h里面

Command【4-3】

public enum class Command {【4-3】

handShakeReq,

handShakeAck,

flashWriteReq,

flashWriteAck,

flashReadReq,

flashReadAck,

flashEraseReq,

flashEraseAck,

nvramWriteReq,

nvramWriteAck,

nvramReadReq,

nvramReadAck,

nvramEraseReq,

nvramEraseAck,

snReq,

snAck,

snConfirmReq,

snConfirmAck,

chipIdReadReq,

chipIdReadAck,

highSecurityReq,

highSecurityAck,

highSecurityConfirmReq,

highSecurityConfirmAck,

platformIdReadReq,

platformIdReadAck,

};

在soc\_packet.cpp里面：

Packet::uIntToByteArray【5】

Int32 Packet::uIntToByteArray(UInt32 uInt, array<Byte>^ \*pArray) {【5】

Int32 errCode = 0;

\*pArray = gcnew array<Byte>(sizeof(uInt));

(\*pArray)[0] = MASK\_FILTER\_VALUE(uInt, MASK\_BYTE\_4);【5-1】【5-2】

(\*pArray)[1] = MASK\_FILTER\_VALUE(uInt, MASK\_BYTE\_3);

(\*pArray)[2] = MASK\_FILTER\_VALUE(uInt, MASK\_BYTE\_2);

(\*pArray)[3] = MASK\_FILTER\_VALUE(uInt, MASK\_BYTE\_1);

return errCode;

}

在soc\_packet.h里面：【5-1】

#define MASK\_BYTE\_4 0xFF000000

#define MASK\_BYTE\_3 0x00FF0000

#define MASK\_BYTE\_2 0x0000FF00

#define MASK\_BYTE\_1 0x000000FF

在soc\_packet.h里面：

MASK\_FILTER\_VALUE

#define MASK\_FILTER\_VALUE(\_X\_, \_MASK\_) (\_X\_ & \_MASK\_) >> Packet::getMaskOffset(\_MASK\_)【5-2】【5-2-1】

在soc\_packet.cpp里面：

getMaskOffset

UInt32 Packet::getMaskOffset(UInt32 mask) { 【5-2-1】

UInt32 offset = 0;

const UInt32 maskBit0 = 0x00000001;

if (mask <= 0) {

return offset;

}

for (offset = 0; offset < sizeof(mask) \* 8; offset++) {

if ((mask & maskBit0) > 0) { // bit0 is 1

break;

}

mask >>= 1;

}

return offset;

}

在soc\_protocol.cpp里面

UartProtocol::write【6】

Int32 UartProtocol::write(array<Byte>^ buffer) {【6】

CHECK\_PORT;【6-1】

#if PACKET\_DETAIL //默认是1表示打印

Packet::print(String::Format("({0}) out (0x{1:X4}) bytes: ", portName, buffer->Length), buffer, 32);

#endif

#if PORT\_READY //默认是1

try {

mSerialPort->Write(buffer, 0, buffer->Length);

} catch (TimeoutException^) {

ConsoleU::writeLine(String::Format("{0} timeout", \_\_FUNCTION\_\_),

ConsoleU::Level::Error);

return -1;

}

#endif

return 0;

}

在soc\_protocol.h里面

CHECK\_PORT;【6-1】

#if PORT\_READY //默认是1，表示打印

#define CHECK\_PORT \【6-1】

do { \

if (mSerialPort == nullptr) { \

ConsoleU::writeLine(String::Format("{0} invalid serial port", \_\_FUNCTION\_\_), ConsoleU::Level::Error); \

return -10; \

} \

if (!mSerialPort->IsOpen) { \

ConsoleU::writeLine(String::Format("({0}) is not opened", mSerialPort->PortName), ConsoleU::Level::Error); \

} \

} while(0)

#else

#define CHECK\_PORT

#endif

在soc\_protocol.cpp里面

UartProtocol::read【7】

Int32 UartProtocol::read(array<Byte>^ buffer) {【7】

Array::Copy(mReadBuffer, 0, buffer, 0, buffer->Length);【7-1】

return buffer->Length;

}

在soc\_protocol.cpp里面 构造函数里面

mReadBuffer = gcnew array<Byte>(READ\_BUFFER\_LENGTH);【7-1】【7-1-1】

在soc\_protocol.h里面

#define READ\_BUFFER\_LENGTH 0x1FFFF // must > 64KB(0xFFFF) 【7-1-1】

在soc\_protocol.cpp里面：

UartProtocol::parse【8】

Int32 UartProtocol::parse(array<Byte>^ packet, Command \*pCmd, array<Byte>^ dataAck)【8】

{CHECK\_IN;【1】

Byte cell = 0;

for (Int32 i=0; i<packet->Length -1; i++) {// check sum

cell += packet[i];

}

if (cell != packet[packet->Length -1]) {

ConsoleU::writeLine(String::Format("check sum({0} != {1}) error", cell.ToString("X2"),

packet[packet->Length -1].ToString("X2")), ConsoleU::Level::Error);

return -2;

}

cell = packet[(Int32) Index::startCode]; // check key cells 【4-1】

if (cell != BYTE\_START\_CODE) {

ConsoleU::writeLine(String::Format("check start code[{0}] error", cell),

ConsoleU::Level::Error);

return -1;

}

cell = packet[(Int32) Index::protocolType];

if (cell != BYTE\_PROTOCOL\_TYPE) {

ConsoleU::writeLine(String::Format("check protocol type[{0}] error", cell),

ConsoleU::Level::Error);

return -1;

}

cell = packet[(Int32) Index::commandDirection];

if (cell != (Byte) Direction::stbSend) {

ConsoleU::writeLine(String::Format("check Direction[{0}] error", cell),

ConsoleU::Level::Error);

return -1;

}

cell = packet[(Int32) Index::commandCode];

if (Enum::IsDefined(Command::typeid, (Int32) cell)) {// output cmd string

\*pCmd = (Command) cell;// IsDefined返回指定枚举中是否存在具有指定值的常数的指示

ConsoleU::writeLine(String::Format("parse cmd=\"{0}\"", ((Command) \*pCmd).ToString()),

ConsoleU::Level::Info);

}

if (dataAck != nullptr) { // output data sector

Array::Copy(packet, (Byte) Index::data, dataAck, 0, packet->Length - CONTAINER\_LENGTH);

}

CHECK\_OUT;【2】

return 0; }

// send address and length

partitionAddr {【9】

在soc\_flash.h里面：

property UInt32 partitionAddr {【9】

UInt32 get(void) {

UInt32 addr = 0;

getAddr(partition, &addr);【9-1】【9-2】

return addr;

}

}

在soc\_flash.cpp里面：

getAddr【9-1】

Int32 Flash::getAddr(Partition p, UInt32 \*pAddr) {【9-1】【9-1-1】

Int32 errCode = 0;

switch (p) {

case Partition::boot:

\*pAddr = mCrtModel->addrBoot;

break;

case Partition::dsp:

\*pAddr = mCrtModel->addrDsp;

break;

case Partition::app:

\*pAddr = mCrtModel->addrApp;

break;

case Partition::fs:

\*pAddr = mCrtModel->addrFs;

break;

default:

break;

}

return errCode;

}

Partition【9-1-1】

在soc\_model.h里面：

public enum class Partition {【9-1-1】

boot,

dsp,

app,

fs

};

在gf\_main.cpp里面：

mFlash->partition【9-2】

Void GF\_Main::comboBox\_Partition\_SelectedIndexChanged(System::Object^ sender, 【9-2】System::EventArgs^ e) {

if (comboBox\_Partition->SelectedItem == nullptr) { return; }

mFlash->partition = (Partition) Enum::Parse(Partition::typeid, comboBox\_Partition->SelectedItem->ToString());

}

CHECK\_CANCEL【10】

在gc\_burn.cpp里面

#define CHECK\_CANCEL \【10】

do { \

if (mWorker->CancellationPending) { \

ConsoleU::writeLine(String::Format("cancel ({0}) burn process", mProtocol->portName), \

ConsoleU::Level::Warning); \

return; \

} \

} while (0) \

Packet::Packet【11】

在soc\_packet.cpp里面：

Packet::Packet(UInt32 addr, UInt32 length) {【11】

mFlashAddr = addr;

mFlashLength = length;

}

UartProtocol::flashWrite【12】

在soc\_protocol.cpp里面：

Int32 UartProtocol::flashWrite(Packet^ packet, Int32 timeout) {【12】

Int32 errCode = 0;

array<Byte>^ dataReq;

errCode = packet->pack(&dataReq);【12-1】

if (errCode < 0) {

return errCode;

}

array<Byte>^ dataAck = gcnew array<Byte>(ACK\_WRITE\_DATA\_LENGTH\_FLASH);//宏是3

if (timeout != 0) {

errCode = writeDrive(Drive::flash, dataReq, dataAck, timeout);【12-2】

} else {

errCode = writeDrive(Drive::flash, dataReq, dataAck);

}

if (errCode < 0) {

return errCode;

}

ReturnCode result = (ReturnCode) dataAck[ACK\_WRITE\_DATA\_RETURN\_INDEX];//宏是2【12-3】

if (result != ReturnCode::ok) {

ConsoleU::writeLine(String::Format("write flash error({0})", result), ConsoleU::Level::Error);

return -3;

}

Int16 iAckPacketNumber = dataAck[0];

iAckPacketNumber = iAckPacketNumber << 8;

iAckPacketNumber += dataAck[1];

if (packet->number != iAckPacketNumber) {【12-4】

ConsoleU::writeLine(String::Format("acknowledge wrong packet number({0:X2})",

iAckPacketNumber), ConsoleU::Level::Warning);

return -1;

}

return errCode;

}

在soc\_packet.h中

packet->pack【12-1】

在soc\_packet.cpp里面：

Int32 Packet::pack(array<Byte>^ \*pPacket) {【12-1】

if (mNumber == 0 && mTotal == 0) {

array<Byte>^ aAddrLeng;

Packet::appendUIntToArray(mFlashAddr, mFlashLength, &aAddrLeng);【12-1-1】

Int32 lengTotal = sizeof(mNumber) + sizeof(mTotal) + aAddrLeng->Length;

\*pPacket = gcnew array<Byte>(lengTotal);

aAddrLeng->CopyTo(\*pPacket, sizeof(mNumber) + sizeof(mTotal));

return 0;

} else if (mTotal != 0) {

array<Byte>^ number;

Packet::uIntToByteArray(mNumber, &number); 【5】

array<Byte>^ total;

Packet::uIntToByteArray(mTotal, &total); 【5】

array<Byte>^ length;

Packet::uIntToByteArray(mLength, &length); 【5】

array<Byte>^ crc;

Packet::uIntToByteArray(mCrc, &crc); 【5】

Int32 lengTotal = number->Length + total->Length + length->Length + mData->Length + crc->Length;

\*pPacket = gcnew array<Byte>(lengTotal);

Int32 index = 0;

number->CopyTo(\*pPacket, index);

index += number->Length;

total->CopyTo(\*pPacket, index);

index += total->Length;

length->CopyTo(\*pPacket, index);

index += length->Length;

mData->CopyTo(\*pPacket, index);

index += mData->Length;

crc->CopyTo(\*pPacket, index);

}

return 0;

}

Packet::appendUIntToArray【12-1-1】

在soc\_packet.cpp里面：

Int32 Packet::appendUIntToArray(UInt32 uA, UInt16 uB, array<Byte>^ \*pArray) {【12-1-1】

Int32 errCode = 0;

array<Byte>^ aA;

errCode = Packet::uIntToByteArray(uA, &aA);【5】

if (errCode < 0) {

return errCode;

}

array<Byte>^ aB;

errCode = Packet::uIntToByteArray(uB, &aB); 【5】

if (errCode < 0) {

return errCode;

}

errCode = Packet::appendArray(aA, aB, pArray); 【5】

if (errCode < 0) {

return errCode;

}

return errCode;

}

UartProtocol::writeDrive【12-2】

在soc\_protocol.cpp里面：

Int32 UartProtocol::writeDrive(Drive drv, array<Byte>^ dataReq, array<Byte>^ dataAck,【12-2】

Int32 timeout) {

Int32 errCode = 0;

CHECK\_IN;

Command cmdReq;

if (drv == Drive::flash) {

cmdReq = Command::flashWriteReq;

} else if (drv == Drive::nvram) {

cmdReq = Command::nvramWriteReq;

}

Command cmdAck;

if (drv == Drive::flash) {

cmdAck = Command::flashWriteAck;

} else if (drv == Drive::nvram) {

cmdAck = Command::nvramWriteAck;

}

errCode = sendCommandData(cmdReq, cmdAck, dataReq, dataAck, timeout);//在主页

if (errCode < 0) {

return errCode;

}

CHECK\_OUT;

return errCode;

}

ReturnCode【12-3】

在soc\_protocol.h里面

public enum class ReturnCode {【12-3】

ok,

failed,

timeout,

};

number【12-4】

property Int32 number {

Int32 get(void) {

return mNumber;

}

}

// send packet

packetTotal【13】

在soc\_flash.h里面

property UInt32 packetTotal {【13】

UInt32 get(void) {

UInt32 total = mBinBuffer->Length / model->sizePacket;【13-1】【13-2】

if (mBinBuffer->Length % model->sizePacket != 0) {

total += 1;

}

return total;

}

}

在soc\_flash.cpp里面

mBinBuffer【13-1】

Int32 Flash::loadBin(String^ fileName) { 【13-1】

Int32 errCode = 0;

if (!File::Exists(fileName)) {

return -1;

}

mBinBuffer = File::ReadAllBytes(fileName);

ConsoleU::writeLine(String::Format("load bin file ({0}), ({1}K) bytes", fileName,

mBinBuffer->Length / 1024), ConsoleU::Level::Normal);

return errCode;

}

sizePacket【13-2】

model->sizePacket在soc\_flash.cpp的loadConfig()方法中初始化赋的值【13-2】

在soc\_flash.cpp中

getPacket【14】

Int32 Flash::getPacket(UInt32 nb, array<Byte>^ \*pPacket) {【14】

Int32 errCode = 0;

CHECK\_BIN;【14-1】

UInt32 offset = nb \* mCrtModel->sizePacket;

UInt32 length = 0;

if (nb < packetTotal -1) {

length = mCrtModel->sizePacket;

} else { // last packet

length = mBinBuffer->Length - (packetTotal -1) \* mCrtModel->sizePacket;

}

\*pPacket = gcnew array<Byte>(length);

Array::Copy(mBinBuffer, offset, \*pPacket, 0, (\*pPacket)->Length);

return errCode;

}

CHECK\_BIN \【14-1】

在soc\_flash.cpp中

#define CHECK\_BIN \【14-1】

do { \

if (mBinBuffer == nullptr || mBinBuffer->Length == 0) { \

ConsoleU::writeLine("haven't load bin file yet", ConsoleU::Level::Error); \

return -1; \

} \

} while (0)

Packet::Packet【15】

在soc\_packet.cpp中：

Packet::Packet(Int16 number, Int16 total, UInt16 length, array<Byte>^ data, UInt32 crc) {【15】

mNumber = number;

mTotal = total;

mLength = length;

mData = data;

mCrc = crc;

}

UartProtocol::flashWrite【16】

在soc\_protocol.cpp中

Int32 UartProtocol::flashWrite(Packet^ packet) {【16】

Int32 errCode = 0;

CHECK\_IN;【1】

errCode = flashWrite(packet, 0);【12】

if (errCode < 0) {

return errCode;

}

CHECK\_OUT;【2】

return errCode;

}

UartProtocol::flashRead【17】

在soc\_protocol.cpp中

Int32 UartProtocol::flashRead(UInt32 addr, array<Byte>^ data) {【17】

return readDrive(Drive::flash, addr, data);【17-1】

}

UartProtocol::readDrive【17-1】

在soc\_protocol.cpp中

Int32 UartProtocol::readDrive(Drive drv, UInt32 addr, array<Byte>^ data) {【17-1】

Int32 errCode = 0;

CHECK\_IN;

array<Byte>^ aDataReq;

errCode = Packet::appendUIntToArray(addr, (UInt16) data->Length, &aDataReq);【12-1-1】

if (errCode < 0) {

return errCode;

}

Command cmdReq;

if (drv == Drive::flash) {

cmdReq = Command::flashReadReq;

} else if (drv == Drive::nvram) {

cmdReq = Command::nvramReadReq;

}

Command cmdAck;

if (drv == Drive::flash) {

cmdAck = Command::flashReadAck;

} else if (drv == Drive::nvram) {

cmdAck = Command::nvramReadAck;

}

Int32 ackDataSectorLength = 2 /\* Length \*/ + data->Length + 4 /\* Crc \*/;

array<Byte>^ aDataAck = gcnew array<Byte>(ackDataSectorLength);

errCode = sendCommandData(cmdReq, cmdAck, aDataReq, aDataAck, mTimeoutRead);//在主页

if (errCode < 0) {

return errCode;

}

Array::Copy(aDataAck, ACK\_READ\_DATA\_INDEX\_DATA, data, 0, data->Length);

// check data length

UInt16 iLeng = aDataAck[0] \* 0x100 + aDataAck[1];

if (iLeng != data->Length) {

ConsoleU::writeLine(String::Format("ack data length is not matched, ({0} != {1})", iLeng, data->Length),

ConsoleU::Level::Error);

return -1;

}

// check crc

UInt32 iCrc = Crc::crc32(data);【17-1-1】

array<Byte>^ aCrc;

Packet::uIntToByteArray(iCrc, &aCrc);【5】

iCrc = BitConverter::ToUInt32(aCrc, 0);

array<Byte>^ aCrcAck = gcnew array<Byte>(4);

Array::Copy(aDataAck, ACK\_READ\_DATA\_INDEX\_DATA + data->Length, aCrcAck, 0, aCrcAck->Length);

Int32 iCrcAck = BitConverter::ToInt32(aCrcAck, 0);

if (iCrc != iCrcAck) {

ConsoleU::writeLine(String::Format("ack data crc is not matched, ({0:X2} != {1:X2})", iCrc, iCrcAck),

ConsoleU::Level::Error);

return -2;

}

CHECK\_OUT;【2】

return 0;

}

Crc::crc32【17-1-1】

在soc\_crc.h中

UInt32 Crc::crc32(array<Byte>^ buffer) {【17-1-1】

UInt32 crc32val = 0;

int i;

for (i = 0; i < buffer->Length; i++) {

crc32val = crc32\_tab[(crc32val ^ buffer[i]) & 0xff] ^ (crc32val >> 8);//也和上面是同页

}

return crc32val;

}