// hand shake

//soc\_protocol.cpp里面

Int32 UartProtocol::handshake(void) {

Int32 errCode = 0;

**CHECK\_IN**;【1】

errCode = **sendCommand**(Command::handShakeReq, Command::handShakeAck, nullptr);

if (errCode < 0) {

return errCode;

}

**CHECK\_OUT;** 【2】

return errCode;

}

Int32 UartProtocol::sendCommand(Command cmdReq, Command cmdAck, array<Byte>^ dataAck) {

Int32 errCode = 0;

**CHECK\_IN;**【1】

errCode = **sendCommandData**(cmdReq, cmdAck, nullptr, dataAck, mTimeoutCmd);

if (errCode < 0) {

return errCode;

}

**CHECK\_OUT;**【2】

return errCode;

}

Int32 UartProtocol::sendCommandData(Command cmdReq, Command cmdAck, array<Byte>^ dataReq,

array<Byte>^ dataAck, Int32 timeout) {

Int32 errCode = 0;

**CHECK\_IN**;【1】

// set bytes want to read before writing

mBytesWantToRead = ((dataAck != nullptr) ? dataAck->Length : 0) + CONTAINER\_LENGTH;

// flush serial port

mSerialPort->DiscardInBuffer();

**REPORT\_CMD**(cmdReq);【3】

array<Byte>^ packet;

errCode = **pack(**cmdReq, dataReq, &packet);【4】

if (errCode < 0) {

return errCode;

}

errCode = **write**(packet);【6】

if (errCode < 0) {

return errCode;

}

if (!mSemaphore->WaitOne(timeout)) {

ConsoleU::writeLine(String::Format("({0}) didn't response after ({1})ms", portName, timeout),

ConsoleU::Level::Error);

return -5; // timeout

}

array<Byte>^ dataR = gcnew array<Byte>(mBytesWantToRead); // read

Int32 numbers = **read(**dataR);【7】

Command cmd;

errCode = **parse**(dataR, &cmd, dataAck);【8】

if (errCode < 0) {

return errCode;

}

if (cmd != cmdAck) {

//throwExceptionAckWrongCommand(cmd);

ConsoleU::writeLine(String::Format("wrong cmd responsed. not ({0}) expected", cmdAck),

ConsoleU::Level::Error);

}

CHECK\_OUT; return errCode; }

// send address and length

在soc\_burn.cpp里面：

UInt32 addr = mFlash->partitionAddr; 【9】

if (errCode < 0) {

ConsoleU::writeLine("get partition addr error", ConsoleU::Level::Error);

return;

}

UInt32 size = mFlash->partitionSize; 【和 9一样的原理】

if (errCode < 0) {

ConsoleU::writeLine("get partition size error", ConsoleU::Level::Error);

return;

}

for (;;) {

CHECK\_CANCEL; 【10】

mWorker->ReportProgress(1);

Packet^ packet = gcnew Packet(addr, size);【11】

errCode = mProtocol->flashWrite(packet, 20000/\* 20s for erase process \*/);【12】

#if BLACKBOX\_TEST

break;

#else

if (errCode == 0) {

break;

}

#endif

}

// send packet

{

UInt32 total = mFlash->packetTotal;【13】

for (UInt32 nb = 0; nb < total; ) {

CHECK\_CANCEL;

// notify progress bar

Int32 percentComplete = (int)((float) (nb +1) / (float) total \* 100);

mWorker->ReportProgress(percentComplete);

array<Byte>^ aWb;

errCode = mFlash->getPacket(nb, &aWb);【14】

if (errCode < 0) {

continue; // try again

}

Packet^ packet = gcnew Packet((Int16) nb, (Int16) total,【15】

(UInt16) aWb->Length, aWb, Crc::crc32(aWb));

DateTime timeStart = DateTime::Now; // start timer

errCode = mProtocol->flashWrite(packet);【16】

#if BLACKBOX\_TEST

#else

if (errCode < 0) {

ConsoleU::writeLine(String::Format("write flash error, packet nb={0:X2}/{1:X2}", nb, total),

ConsoleU::Level::Error);

if (mIsErrorTerminate) {

return; // terminated

}

continue; // try again

}

#endif

// check

if (checkBox\_Check->Checked) {

Boolean errorOccured = false;

do {

errorOccured = false;

array<Byte>^ aRb = gcnew array<Byte>(aWb->Length);

UInt32 addr = mFlash->partitionAddr + nb \* mFlash->model->sizePacket;

errCode = mProtocol->flashRead(addr, aRb);【17】

if (errCode < 0) {

errorOccured = true; // try again

continue;

}

for (Int32 i = 0; i < aWb->Length; i++) {

if (aWb[i] != aRb[i]) {

ConsoleU::writeLine(String::Format("{0} packet check failed, address=0x{1:X2}, write(0x{2:X2})!=read(0x{3:X2})",

nb, addr + i, aWb[i], aRb[i]), ConsoleU::Level::Error);

errorOccured = true;

}

}

} while (errorOccured);

}

// calculate time remain

DateTime timeEnd = DateTime::Now;

mTimeSpan = TimeSpan::FromTicks((timeEnd.Ticks - timeStart.Ticks) \* (total - nb -1));

nb++; // process next packet

}

}