

protocol

Data transfer has only one package (reading/writing) transfer for functions that do not require big data amount (everything except frames reading and flash reading/writing)

```
getStatus
{
To device:
outReport[0] = 0
outReport[1] = 1;
from device:
inReport[0]=0x81;
inReport[1]=flags;
inReport[2]=LO(framesInMemory);
inReport[3]=HI(framesInMemory);
}
setExposure
{
To Device:
outReport[1]=2;
outReport[2]=exposure[0]; //low byte
outReport[3]=exposure[1]; //little endian!
outReport[4]=exposure[2];
outReport[5]=exposure[3]; //high byte
outReport[6]=force;
From Device:
inReport[0]=0x82;
inReport[1]=errorCode;
}
setAcquisitionParams
{
To Device:
outReport[0]=3;
outReport[1]=LO(scans);
outReport[2]=HI(scans);
outReport[3]=LO(blankScans);
outReport[4]=HI(blankScans);
outReport[5]=continuousScanMode;
from device:
inReport[0]=0x83;
inReport[1]=errorCode;
}
setFrameFormat
{
to device:
outReport[0]=4;
outReport[1]=LO(startElement);
```

```

outReport[2]=HI(startElement);
outReport[3]=LO(endElement);
outReport[4]=HI(endElement);
outReport[5]=reduce;
from device:
inReport[0]=0x84;
inReport[1]=errorCode;
inReport[2]=LO(frameElements);
inReport[3]=HI(frameElements);
}
setExternalTrigger
{
to device:
outReport[0]=5;
outReport[1]=triggerEnabled;
outReport[2]=triggerFront;
from device:
inReport[0]=0x85;
inReport[1]=errorCode;
}
softwareTrigger (start accumulation in any case)
{
to device:
outReport[0]=6;
from device: nothing
}
clearMemory
{
to device:
outReport[0]=7;
from device:
inReport[0]=0x87;
inReport[1]=errorCode;
}
getFrameFormat
{
to device:
outReport[0]=8;
from device:
inReport[0]=0x88;
inReport[1]=LO(startElement);
inReport[2]=HI(startElement);
inReport[3]=LO(endElement);
inReport[4]=HI(endElement);
inReport[5]=redux;
inReport[6]=LO(frameElements);
inReport[7]=HI(frameElements);
}

```

Frame transfer

getFrameElements

Packets request (frame packets) starts from **offset** (element number, not byte number).

Device replies with several packets defined in **Packets**, offset is incremented automatically and points to current offset relative to the beginning of the frame
packetsRemaining_or_IsError shows how many packets will be transferred

Equal to 0 if it is last packet

if ≥ 250 means error and data cannot be considered as correct.

```
{
to device:
outReport[0]=0x0A;
outReport[1]=LO(offset);
outReport[2]=HI(offset);
outReport[3]=LO(frameNum);
outReport[4]=HI(frameNum);
outReport[5]=packets;
from device:
packets:
inReport[0]=0x8A;
inReport[1]=LO(offset);
inReport[2]=HI(offset);
inReport[3]=packetsRemaining_or_IsError;
inReport[4]=LO( frame[offset]);
inReport[5]=HI( frame[offset]);
inReport[6]=LO( frame[offset+1]);
inReport[7]=HI( frame[offset+1]);
...
inReport[62]=LO( frame[offset+29]);
inReport[63]=HI( frame[offset+29]); //elements outside the frame are filled with 0
or random numbers
}
userFlashErase
{
to device:
outReport[0]=0x1C;
from device:
inReport[0]=0x9C;
inReport[1]=errorCode;
}
userFlashWrite
{
to device:
outReport[0]=0x1B;
outReport[1]=offset[0]; //low byte
outReport[2]=offset[1]; //2nd
outReport[3]=offset[2]; //3rd
outReport[4]=offset[3]; //high byte
```

```

outReport[5]=numberOfbytes; // <= 58 (max payload length)
outReport[6]=bytesToWrite[0]; //payload begin
outReport[7]=bytesToWrite[1];
...
outReport[n]=bytesToWrite[m]; //payload end
//n=numberOfbytes+5, m=numberOfbytes1;
from device:
inReport[0]=0x9B;
inReport[1]=errorCode;
}
Maximum number of packages = 100
userFlashRead //works almost in the same way as getFrameElements difference: offset in
request is absolute. Offset in reply is relative.
{
to device:
outReport[0]=0x1A;
outReport[1]=absoluteOffset[0]; //low byte
outReport[2]=absoluteOffset[1]; //2nd
outReport[3]=absoluteOffset[2]; //3rd
outReport[4]=absoluteOffset[3]; //high byte
outReport[5]=packets;
from device:
packets:
inReport[0]=0x9A;
inReport[1]=LO(localOffset);
inReport[2]=HI(localOffset);
inReport[3]=packetsRemaining_or_IsError;
inReport[4]=flash[absoluteOffset+localOffset];
inReport[5]=flash[absoluteOffset+localOffset+1];
..
inReport[63]=flash[absoluteOffset+localOffset+59];
}
setOpticalTrigger
{
to device:
outReport[0]=0x0B;
outReport[1]=triggerMode;
outReport[2]=LO(pixel);
outReport[3]=HI(pixel);
outReport[4]=LO(threshold);
outReport[5]=HI(threshold);
from device:
inReport[0]=0x8B;
inReport[1]=errorCode;
}

```