1 Introduction

The Apogee Alta camera drivers are ActiveX drivers. It turned out to be hard to make them work under Borland C++. The data acquisition program Vision is written in Borland C++ and needs access to the Alta camera. The solution to this problem is ApogeeServer. It is a small Visual C++ program which communicates on one side with the Alta camera using the ActiveX drivers delivered with the camera and on the other side with Vision over ethernet using TCP/IP. The program can easily be modified to support other camera systems than the Apogee Alta and make these cameras accessible to Vision as long as Visual C++ drivers are delivered with the camera.

2 Installation of the Alta system

The Alta comes delivered with drivers. Call the install batch file delivered with them. Sometimes the batch file doesn't work correctly and you have to manually do what it intends to do: copying the DLL and registering it with regsvr32. Power the Alta up. Wait until the LEDs blink! Hook the Alta up to your system. Windows asks for some drivers to be installed. Use Maxim, which is delivered with the Alta to test if it works. Eventually switching the Alta off and on, rebooting and verifying that the DLL and the .SYS file are at the correct place and registered should make everything work.

Probably you need to fabricate an external trigger cable. It is a 8pin mini din mal plug, like the Apples serial port connector. The external trigger has to be hooked up on pin 1 and its ground to pin 8.

```
I/O for Alta
----------------------------
Pin 1 - Trigger Input
Pin 2 - Shutter Output
Pin 3 - ShutterStrobe Output
Pin 4 - External Shutter Input
Pin 5 - External Readout Input
Pin 6 - Timer Pulse Input
Pin 7 is 12V from the camera head input
Pin 8 is ground.

 Connector pin layout:

      1 2
     3 4   5
      6 7 8

 There is a bigger gap between
 pin 4 and 5 than anywhere else.
 Below pin 7 and to the sides of
 pin 1 and 2 are mechanical markers
 helping to orient the plug correctly
```

3 User interface

The user interface displays the system status in a textbox to the left and has menu options on buttons to the right. The "Expose" button starts and internally triggered exposure. The "IO setup", "LED setup", "Temperature" and "Search" buttons call the drivers corresponding dialog boxes. The "Test trigger" button switches the trigger line to be an output and toggles it a few times slowly from low to high to test the external trigger cable. The "Expose Ext" button starts an externally triggered exposure". The "Debug start" button starts registering the TCP/IP communication in the file "c: Debug.dat". "Debug stop" stops this. "Exit" quits the program.

4 Communication with Vision

Here is a transcript of the TCP/IP communication with Vision. The first column is the time of communication in milliseconds, the second the direction ("¡¡" means from Vision to Apogeeserver) and the third is the text actually send. In fact each text send starts with "*" and ends with "#", but these markers have been left out ini this protocol.

```
5902296 << Parameters
5902437 << 0002
5902437 << 0002
5902437 << 0004
5902437 << 1019
5902437 << 0004
5902437 << 1019
5902437 << 0020
5902437 << -030
5902437 << 0001
5902437 << 0000
5903843 << Image
5903968 << 0003
5914484 >> Ready
5914484 << Ready
5914500 >> 516128
5914500 >> SendData 516128
5914718 << Ready
5914718 >> 516128
5914718 >> SendData 516128
5914937 << Ready
5914937 >> 516128
5914953 >> SendData 516128
```

The first command "Parameters" tells Apogeeserver that the image parameters will be send. They follow next in this order: BinningX, BinningY, XMin, XMax, YMin, YMax, exposure time in milliseconds, goal temperature of CCD, external trigger (1=external, 0=internal), and the wait time between images in a series of images to be taken.

The next command "Image" tells Apogeeserver to take Images. The number of images to be taken is send as parameter. Apogeeserver answers with "Ready". Vision tells Apogeeserver with "Ready" that it is ready to receive data. Apogeeserver then send the number of bytes in the picture and then the data of the pixture.

5 Classes

The application is initialized in CApogeeServer::InitInstance(). If you want to change the IP port (normally 703) you do it here. The camera is represented by CApogee and initialized with CApogee::InitCamera(). Here you can modify the startup behavior eg. define what the LEDs do or switch off the fans. The user dialog is CApogeeServerDlg and the TCP/IP interface is CTCPIPServer.

6 Events

Two sorts of events can trigger something to happen. Either the user pushes a button. For example the "Expose" button calls CApogeeServerDlg::OnExpose(). This method then calls the appropriate functions of CApogee. The other is that Vision sends a command. They are received in CTCPIPServer::ProcessMessage(). In dependance of the message functions of CTCPIPServer executing the message are called. You can add new commands here if you need to. Look how CTCPIPServer::SetParameters or TakeImage communicate with Vision and CApogee to learn how to add new functionality. Essentially you read parameters from Vision using the ReadInt function and you pass those parameters on to CApogee.

7 Special remarques

Our Alta camera driver has a problem accepting 0 as value for yMin. It always converts it to 1. Combined with binning this makes it necessary for us to start not with line 0 but line 4. The Alta driver also sometimes stalled after an image with binning. That's what the ResetSystem() commands are for. They avoid this situation to occur.