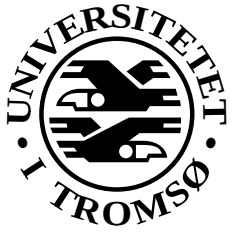
Time Resolved CCD Photometry

Development of a multi-windowing system
and tests on variable subdwarf stars

A dissertation for the degree *doctor scientiarum*
by

Roy H. Østensen

November, 2000



University of Tromsø
Faculty of Science
Department of Physics
Astrophysics Group

Preface

Work on this thesis was originally initiated in the autumn of 1994, while I was visiting the Astronomical Observatory of Copenhagen University (CUO) at Brorfelde. My stay at Brorfelde, located in the countryside about 80 kilometers west of Copenhagen, was financed by a mobility grant from the Nordic Academy of Science (NorFA). At this idyllic place I was working with Ralph Florentin-Nielsen on a follow-up study after my Master project on gravitational lenses, extending our photometric light curves of the QSO 2237+305 with the most recent data and preparing the results into a final paper [OEs96]. At first, I wanted to continue the work on gravitational lenses into a Ph.D. project, keeping up the collaboration with Rolf Stabell and Sjur Refsdal at the Institute of Theoretical Astrophysics of the University of Oslo, who had been external supervisors for my Master project, as well as with Ralph Florentin-Nielsen and Jan Teuber, who had been close collaborators on the project. Unfortunately, the Department of Physics in Tromsø refused to approve such a project since it was too loosely attached to the established scientific activities of the department. Thus, Jan-Erik Solheim drafted a new plan for a Ph.D. project that conformed with the requirements of the Department.

This project involved the development, construction and building of a CCD photometer for continuous photometry, as well as testing of this instrument on the pulsating central stars in planetary nebulae. The new project was inspired by the CUO's recent achievements in designing and building CCD cameras, primarily for the Nordic Optical Telescope, under the leadership of Ralph Florentin-Nielsen. A requirement for the project was a CCD-controller built at CUO and based on the existing design, and mechanical camera parts built at the Department of Physics in Tromsø.

In a strike of misfortune, Florentin-Nielsen became seriously ill during the spring of 1995, and passed away in October the same year. Naturally, this created significant problems for the small development team in Copenhagen, and I found my self for a while considering options for a new Ph.D. project. For the following years I worked at the Nordic Optical Telescope (NOT) and at the Institut d'Astrophysique in Liège, Belgium, working from time to time on software solutions applicable for the CCD photometry project, in between all the other work. In the autumn of 1998 we received signals from the CUO that they were getting ready to complete

and deliver our controller electronics in the nearest future, and my optimism for the project returned. I then started writing a thorough description of the hardware system, which is what became Chapter 2 of this work. The control code for the CCD camera controller electronics is described in Chapter 3, and the user interface programs in Chapter 4.

The CCD controller was delivered in March 1999, and I proceeded to get the windowed readout algorithms up and running in only two months time, using an engineering grade CCD chip kindly lent to us by the CUO. The chip was installed in the camera head, built at our mechanical workshop in Tromsø. After initial testing of the windowed readout code in Tromsø, the system was brought to La Palma in early April 1999 and successfully tested during technical time at the NOT, on May 1–2. The tests proved to be good enough, not only to give us confidence that the system was reliable, but to be of scientific value. A description of these results are given in Chapter 5.

The camera's filter and shutter unit was built at the workshop in Tromsøduring the summer of 1999, and I wrote and tested the control system and interface to this unit during the early autumn. The complete system was tested in Skibotn during the autumn and winter, 1999–2000, a work conducted primarily by Master students Robert Kamben and Frank Johannessen. The light curves Kamben collected for his Master thesis has proven not only that the system worked as intended, but that we can perform reliable time resolved photometry during Auroral conditions; a feat classical photometric detectors never managed to do. The results of the test of the complete system are therefore not to be found in this work, but will be presented in the forthcoming Master thesis of Robert Kamben.

After our CCD system had been completed and tested, I decided to convert the windowed readout programs to work on the High Resolution Adaptive Camera at the NOT, for an observation run in October 1999. Our objective was now to identify new pulsating stars of the subdwarf B type, a completely new type of variable star discovered in 1997. The excellent results and discoveries made during these five nights are presented in Chapter 6.

I wish to acknowledge my gratitude to all the people who has contributed to this work. First of all, special thanks to my supervisor Jan-Erik Solheim, who has done his very best to support my work and has dug deeply to provide me with financial aid during the hardest times, in spite of the repeated rejections from the Norwegian Research Council.

Due thanks are given to Preben Nørregaard, Jens Klougart and the rest of the staff at the CUO, for all their great work and helping me putting things together. Credits are also given to the staff at the Nordic Optical Telescope, for letting me try out my ideas. Special thanks are given to Torfinn Roaldsen and Yngve Eilertsen at the University of Tromsø's Department of Physics, whose enthusiastic work on the electronic and mechanical parts of has been essential for this project. Thanks

also to our departments associated professor Edmund Meištas has also contributed significantly to this project by his original mechanical designs. Many thanks also to Robert Kamben and Frank Johannessen who have given a great contribution to showing that this thing actually works. I also wish to thank the people at Moletai Observatory whose interest in this project has been a good support and motivation.

The most special thanks must without question go to my own very special Agnieszka Bronowska, who has been the major motivating factor for me to actually getting my act together and completing this project.

A special *in memoriam* dedication goes to Ralph Florentin-Nielsen, who not only inspired and initiated this project, but whose spirit and memory made it come through.

Inspiration for this project has also come from other sources. I wish to give some additional *in memoriam* dedications to Joachim Nilsen and Mark Sandman, two giant stars who has now joined St. Hendrix and St. Morrison for the eternal show. I'd also like to express my gratitude to Lou Reed, Patti Smith, Nick Cave and all the other eternal stars for impregnating my head with those good vibrations, and hanging on for another year.

O we will know, won't we?
The stars will explode in the sky
O but they don't, do they?
Stars have their moment and then they die

– Nick Cave

Roy Østensen
Tromsø, November 2000

Contents

Chapter 1

Seeing the Light

Time series photometry is the fine art of measuring the changes in brightness of astronomical objects, and producing a reliable light curve for each target. In these light curves all systematic errors due to instrumentation, variable sky background and atmospheric transparency should have been removed. Such light curves can be further analysed for periodic patterns which can reveal important physical parameters for the objects in question. For stellar targets such observations have revealed a wide range of oscillation phenomena with frequencies ranging from seconds to years.

In this introductory chapter I will discuss the detectors that have been traditionally used for time series photometry, up to the state-of-the-art charge coupled device (CCD) systems that have been slowly taking over the optical astronomy arena since the 1980's.

1.1 Classical Photometry

The brightness of an astronomical target is traditionally measured in the magnitude system, a system originating with the ancient Greeks in a time when the only available detector was the human eye. Mathematically, magnitudes are related to flux values by a logarithmic relationship,

$$m = m_0 - 2.5 \log(I), \tag{1.1}$$

where m_0 is the zero-point of the magnitude scale; by definition the brightness of the star Vega.

While the first astronomic revolution introduced by Galileo's invention of the telescope opened up the sky for much deeper scrutiny, the study of variable stars came much later. Since most variable stars show quite modest brightness variations, up to the middle of the 19^{th} century, only a handful of periodically variable stars were

known. This situation changed rapidly with the photographic revolution. As photographic plates allowed permanent recordings of stellar brightnesses, much more detailed and accurate comparisions of stars over long periods of time became possible. The 20^{th} century saw several more revolutions in optical astronomical detectors, most significantly the extremely sensitive and precise photomultiplier tubes that could provide excellent brightness measurements with accurate time resolution, and later the CCD chip with its extremely good quantum efficiency coupled with good imaging capabilities.

A most considerable source of uncertainty, when measuring stellar brightnesses with any modern detector, arises when the zero point of the observation in question must be established. For that reason it is common to only consider *differential photometry* when extremely high precision is desired; in which

$$\Delta m = m - m'. \tag{1.2}$$

Here, both magnitudes are instrumental and, thus, errors from zero pointing are eliminated. When both are measured with the same telescope and detector under fully identical conditions, extremely high precision can be achieved.

Theoretically, such measurements are limited by two intrinsic noise sources; scintillation noise and photon shot noise. Scintillation is caused by atmospheric turbulence along the line of sight, and can be expressed as [You67]:

$$\frac{\Delta I}{I} = S_0 D^{-2/3} X^{3/2} e^{-h/8\mathrm{km}} \Delta t^{-1/2}, \tag{1.3}$$

where D and h is the diameter (in cm) and altitude of the telescope, X is the airmass and Δt is the integration time. The S_0 term is 0.09 for good seeing conditions, but much higher values have been experienced, especially during dusty conditions. If we evaluate the scintillation noise for the Nordic Optical Telescope (NOT) ($D = 2.56$, $h = 2382\,\mathrm{m}$) and typical values $\Delta t = 20\,\mathrm{s}$, $X = 1.3$ we find the error in the differential magnitude

$$< \Delta m > \simeq 1.086 \frac{\Delta I}{I} = 0.6 \text{ mmag}. \tag{1.4}$$

Thus scintillation noise gives a significant limitation on the precision that can be obtained in photometric measurements from the ground.

While scintillation affects all stars independently of their brightness, photon shot noise (counting statistics) varies with the inverse square root of number of photons detected. The r.m.s. intensity variation in photon noise can be expressed simply as [War88, p. 15]

$$\frac{\Delta I}{I} = n^{-1/2}. \tag{1.5}$$

Since the number of photons n that can be detected by an astronomical telescope is proportional to the exposure time Δt, to the *equivalent width* $\Delta\lambda$ of the combined

transmission of filter, detector and telescope, to the square of the aperture D, and to the apparent visual magnitude V through $10^{0.2V}$, the photon shot noise can be expressed as [Kje92]

$$\frac{\Delta I}{I} = D^{-1}\Delta\lambda^{3/2}\Delta t^{-1/2}10^{0.2V} \tag{1.6}$$

Thus, for faint stars the photon shot noise can become a significant term, and may dominate over scintillation noise.

A thorough discussion on scintillation noise, and other noise sources that affect all classical time resolved photometry, can be found in Warner's Chapter 1 [War88].

1.2 Photoelectric photometry

The photomultiplier tube (PMT) is a vacuum tube containing a series of electrodes with high voltages between. The first, positive electrode (dynode) is photosensitive, and releases photoelectrons when illuminated. These are accelerated between the layers, liberating more electrons in a cascade effect for each electrode they pass, creating a measurable current at the final negative electrode (anode). There is no threshold to detection, so single quanta are detected, although the statistical probability that a cascade series is initiated (the quantum efficiency, QE) depends on the material of the photoelectrode and lies between 0.1 and 0.4.

1.2.1 Single-channel Photoelectric Photometry

The classical detector for time resolved photometry is the single-channel photometer. These instruments is applied by cycling through a number of targets; *e.g.* one program star, one or two reference stars and a sky background field for each cycle throughout the observation run. The resulting observations can be interpolated and the differential magnitudes computed to produce a light curve of the program star. This procedure has many advantages. It is easy to find good reference stars on the wide area of sky accessible from the observatory during the observation run, and extinction corrections can be computed relatively easily. All measurements are made with the same detector, so no error from differing sensitivity or individual noise characteristics of different sensors and amplifier chains are introduced. On the other hand, many problems are present in such systems. A significant amount of observation time is spent observing sky and reference targets instead of the target of interest, resulting in what is known as a low *duty cycle*. Further time is wasted moving the telescope between the targets. High demands are set on the telescope's mechanical ability to move accurately and quickly between targets.

Provided that the comparison star is good (not variable and comparable in brightness and color to the program star) the main noise in single-channel photometric

observations will be instrumental or come from the reductions. Instrumental noise is introduced to the time series when the telescope fails to reproduce celestial positions during the night (tracking errors), when the telescope focus drifts off during the night due to changes in mirror temperature or changes in elevation angle, or when the photometer itself suffers from instabilities (gain drift). Reduction noise arises mainly from the extinction corrections, the interpolation of data points (required since observations of the target object, comparison star and background field are not simultaneous), sky background removal, irregularly varying extinction (often caused by atmospheric dust) and second order extinction effects.

It is possible in single-channel photometry to produce quite good extinction correction and sky level removal, but the more accurately one wants to measure these variables, the less time one can spend integrating the program star. This means that the duty cycle will be progressively lower, and that the interpolation errors will increase. Single channel photometry will not produce useful data if the weather conditions are less than excellent, since even the faintest cirrus clouds will introduce extinction variations that cannot be corrected for. In general, there is no way to put together a single-channel instrument that can operate near the theoretical limit given by scintillation and shot noise.

1.2.2 Multi-channel Photoelectric Photometry

To avoid the low duty cycle of the one-channel photometer, and thereby reaching fainter astronomical targets, most observers prefer multi-channel instruments. Such instruments are basically several (usually two or three) separate photometers mounted together in a mechanical construction that allows the the secondary channels to move independently of the fixed primary channel within the telescopes field of view. This system allows for continuous photometry of each channel, giving a duty cycle of 100%, provided that a reference star can be found in the instruments field of view.

Multi-channel photometry has several other advantages over the single-channel design, beside the perfect duty cycle. With three channels it is possible to have simultaneous measurements of the target object, comparison star and background field, thus avoiding interpolation errors. With better measurements of the stable reference star throughout the night, extinction corrections also improve. Unfortunately, the complicated design of such three-channel photometers introduces new problems. With three channels to keep track of, the alignment of the apertures becomes quite an art. And on telescopes with Alt–Az mounting, not only must the tracking be accurate, but the rotation of the instrument must be equally precise during the observation run.

Furthermore, each channel has its own complete set of photomultiplier tube, apertures, filter wheel, amplifier and converter chains. This means that even if the

channels are of identical design, they will have different sensitivity, noise and instrumental drift, and this will introduce errors that are not seen in the one-channel design. This sets a lower limit to the accuracy of multi-channel photometry based on the precision that individual photomultiplier tubes, electronics and optics for the different channels can be built of around one mmag per minute [Kje92].

1.2.3 Instrument development in Tromsø

Several photometers have been built at the department of physics in Tromsø. A one-channel chopping photometer was built and used for testing of photometry in the Auroral zone at Skibotn Observatory [Myr79, Myr80]. Later, a two-channel photometer based on the Texas design was built for the Skibotn Telescope, and was later upgraded to three channels to keep up with the Whole Earth Telescope (WET) standard specifications [Kle95]. The instrument was later used at the Nordic Optical Telescope as one of its first instruments.

Later, a completely new, compact and lightweight three-channel design known as the *Pancake photometer* was constructed in collaboration with scientists at Vilnius University in Lithuania [Mei93b, Mei93c]. It was designed as an easily transportable one-suitcase instrument for campaign observations at remote observatories, and has sucessfully been used for most campaigns of the Whole Earth Telescope since its commisioning.

1.2.4 The Whole Earth Telescope

The Whole Earth Telescope (WET) is an observing network of small to medium sized optical telescopes, distributed in in longitude around the Earth to be able to continuously monitor variable stars without interruption by daylight. The motivation for such coordinated observations is the great improvement in the quality of the temporal spectra produced by a light curve without periodic gaps [Nat90]. Observations from a single site inevitably gets periodic gaps in the light curves due to daylight. These interruptions produce *spectral leakage* in the Fourier transforms of the light curves, which results in a single periodic pulsation not appearing as a single peak in the spectrum, but as a set of multiple peaks known as an *alias pattern* or *alias forest*. The shape of such alias patterns can easily be produced by taking the Fourier transform of a single sine wave with sampling and interruptions as given for a series of observations, and is commonly known as the *spectral window* of the observations. Such alias patterns can make it completely impossible to resolve closely spaced peaks of multi-periodic variable stars, regardless of how many nights of observations are spent to improve the signal to noise of the data.

WET observation campaigns are organised once or twice a year, and typically lasts for two to three weeks. The first were held in March 1988, and this autumn will

see the 20^{th} campaign. One target is selected for each campaign (with a couple of secondary targets), and the control center (HQ), usually at Ames in Iowa, US, coordinates the observations, making sure that one observatory is always on the primary target, whenever the weather conditions allow, and distributing secondary targets when there is a redundancy. Since such campaigns naturally require considerable resources and personnel to operate between ten and twenty observatories around the world, and only two such campaigns are held per year, the HQ has an important task in making the best possible use of the available observation time. For the HQ to be able to make target allocation decisions for the available observatories, they need to make frequent examinations of the quality of the temporal spectrum obtained at any time during the campaign. This is accomplished by the observers at all the different observatories submitting their photometric data by e-mail as soon as one observation run is completed. As Nather *et al.* states [Nat90]:

> To allocate our observing resources in a reasonable way, we must make decisions on practically an hour-by-hour basis around the clock, based on possible overlapping coverage, on weather reports from individual sites, and, to a great extent on the perceived importance of the scientific results pouring in through international electronic mail networks. This means that the data reduction and preliminary Fourier spectral analysis must keep pace.

1.2.5 The WET Software Suite

The real-time data assessment requirement has guided the development of R.E. Nather's software suite for the WET. The data acquisition program is called `Quilt 9` and the data reduction program `QED`, and both runs on IBM PC compatible computers under MS-DOS [Cle93]. The `Quilt 9` program is used to set up the observations and receive data from the photometers amplifier/discriminators. The light curves are shown propagating in real-time on a scalable display, and every measurement are saved to disk in a simple formatted text file. After the observations are completed, the `QED` program can be used to read the files produced by `Quilt 9` and mark data points which represent calibration data in the light curves, points spoiled by clouds, artificial light and so on. Then data reduction proceeds by making a correction for dead-time loss, computing and subtracting a sky level based on the available sky data, and performing the extinction correction. After interpolating over the points flagged as bad or calibration points, the processed data is written to a new file. The data should also be normalised by the mean, and then the mean should be subtracted to produce the *normalised* light curve.

When a clean, normalised light curve has been obtained, it is time to start the Fourier analysis process that will reveal any periodicity in the signal [Kep93]. A special program has been developed by Carl Hansen that uses an optimised version

of the DFT algorithm on each uninterrupted observation run and then adds the Fourier components together in complex space to produce a good spectrum of several interrupted observation runs [Nat90]. This program is known as `NEWSFT`, or its slightly updated translation from Fortran to C, `qsft`.

Even in the best WET runs, the final light curve contains gaps which produce some aliasing of the spectral peaks. To reveal the weaker pulsations in the temporal spectrum it is customary to apply a cleaning algorithm to remove the dominant peaks and alias features introduced by the main periods. This is accomplished by computing synthetic, noiseless light curves based on the main periods and phases found in the spectrum and subtract this from the data set, a process known as *prewhitening* of the data. When a new FT is computed from the prewhitened data the main peaks as well as their alias patterns are removed, and new periods can be revealed. This process is repeated until no more significant peaks can be found. A program called `prewhite` has been written by S.O. Kepler for this purpose, taking a light curve of the same form as for `qsft` as input, requesting a number of periods with corresponding amplitudes and phases, computing an artificial sine curve with the same times as the input data, and subtracting this [Kep93].

Other algorithms have been attempted to handle the problem of removing alias patterns from the main frequencies in order to separate the real frequencies from artifacts of the spectral window. One example is the Clean algorithm [Rob87], and other more advanced deconvolution algorithms have also been attempted (see for instance [Roq00]).

1.3 The CCD Revolution

Since the first astronomical CCD instruments became operational in the 1970's, they have completely revolutionised astronomical observations at the large observatories around the world. The traditional way of measuring magnitudes from photographic emulsions was mainstream in astronomy since the introduction of photography in the ninetieth century, but has during the last decade become completely obsolete. Only a few telescopes, mostly Schmidt-class, are still depending on photographic emulsion, to utilize the large field size of these special telescopes, but even here large mosaic CCD detectors are taking over.

CCD cameras have long since outperformed all competitors in the home video market, with compact designs that fits in the palm of your hand. Now, CCD cameras are starting to take their share of the still photo market also, thanks to larger CCDs with better resolution and color. However, the CCD chips used in these devices are poorly suited for astronomy. The problems include low field coverage, low sensitivity at short wavelengths, high dark current or expensive cooling requirements, and some have special pixel designs for color imaging that make them unsuitable for astrophotometry.

The high precision demands from astrophysicists have encouraged the development of special designs to boost the quantum efficiency, especially in the blue part of the spectrum (thinning), cryostatic cooling systems that practically eliminate dark current, and amplifier designs and readout algorithms that reduce readout noise to the minimum set by the CCD chip's output gate.

Common imaging systems use a technique called *frame transfer* to eliminate the dead time between subsequent integration by utilising half of the CCD image array as a frame store. This part is covered by a non-transparent mask, and after an integration is complete the image is very quickly shifted from the imaging half to the frame store, and normal readout is performed from the covered half while a new image is integrating on the other half. With such a system one may easily achieve a duty cycle close to 100% with 10 second integrations, but half of the area of the CCD has to be covered by a light blocking mask. Although this can be an acceptable sacrifice for most observers interested only in high speed photometry of variable stars, for imaging astronomers this is not a popular idea, since field coverage is in many cases the most important factor. Also, with a frame transfer system, one cannot reach cycle times shorter than the readout time for half the chip without applying further tricks to reduce the number of pixels sampled.

One such trick that can also be applied to standard imaging systems is to align the target star and a reference star along the same line on the CCD, by rotation of the field, and only read out a narrow strip with one star in each end, using the area between the stars for sky background determination. This can also be an efficient method, but in most cases it will limit the number of reference stars to one.

In the system that will be described in this work we have chosen a more sophisticated approach. By redesigning the algorithms that the CCD controller electronics use when reading out the image on the CCD chip, we can limit the time spent on charge conversion to those areas on the chip that we are interested in. In this way we significantly reduce the read-out time, so that the duty cycle fraction increases to an acceptable level. This method is called *windowing*, since the observer can define some square or rectangular areas in the field of view – *windows* – that will be sampled while the rest of the field remains unsampled and blocked from view.

Our primary reason for the choice of the windowing method is its flexibility; the whole imaging area can be used without any special alignment or other tricks, there is no limit to the number of channels that can be defined, and no inherent constraints that reduce the imaging capabilities of the system. The last point has been especially vital because it allows us to run this system on existing instruments at the Nordic Optical Telescope (NOT) without making physical changes to the instruments – an obstruction of the kind required for frame transfer would certainly would not have been accepted for such general purpose instruments.

1.3.1 Dark current and cooling

The dark current in a CCD detector, N_D, can be expressed as a function of temperature T [McL89, p. 119]

$$N_D(T) = \frac{I_0 d^2}{q_{e^-}} e^{\frac{-(T_0-T)}{9.96}} \text{ e}^-/\text{s} \tag{1.7}$$

where I_0 is the dark current at room temperature T_0, d is the pixel size and q_{e^-} is the charge of the electron. CCD temperature is a tradeoff between dark current and quantum efficiency (QE), and modern CCD systems are kept at a regulated temperature fixed at some value, typically -100°C.

For this reason, any CCD camera intended for deep imaging of faint astronomical objects, requires a cryostatic cooling device, where the CCD chip itself resides in a vacuum chamber inside a liquid nitrogen dewar. Such designs are expensive, bulky and requires frequent refilling of LN_2. Peltier-type thermoelectric cooling devices are capable of reaching temperatures down to about $-50°$, and are much simpler to design and operate. The higher operating temperature means that the dark current can become the dominating noise source after only about a minute of observation, and are thus only useful for systems where deep imaging is not an issue.

A significant part of the dark current is generated at the junction between the silicon and the silicon dioxide insulation layer [McL89, p. 121]. Dark current can be significantly reduced by operating the CCD in *inversion* mode, in which the operating voltages of the collecting phase are adjusted in such a way as not to populate the interface layer with holes that minimise dark current generation. Most new CCD chips are manufactured with a technology called multi-pinned phase (MPP) which allows the CCD to operate in inversion mode at all times.

1.3.2 Pixel sampling

In astronomical CCD systems the measurement of each pixel value is performed in particular way known as *correlated double sampling*, or CDS for short, that eliminates noise introduced when reseting the CCD output amplifier. The reset noise is considerable even at cryogenic temperatures, typically between 80 and 200 electrons for common CCDs, and would completely dominate all other noise sources in an astronomical CCD system, if it was not eliminated by the CDS method. The reset noise is introduced when the output capacitor (output gate) is recharged to a fixed reference voltage by switching a reset transistor, between the sampling of subsequent CCD pixel values. During CDS the voltage level of the output gate is measured by the A/D converter both before and after the charge collected by a CCD pixel is shifted onto the output gate. The difference between these two measurements is independent of the reset noise, since the noise is correlated in both measurements. This is the basis of the CDS technique that makes it possible to reach readout noise levels below 10 electrons in astronomical CCD devices.

1.3.3 CCD data reduction

Before the data values that are output from the CCD sampling process can be used to measure temporal and spatial light variations, some corrections known as *preprocessing* is necessary. This preprocessing can be divided into two terms for each pixel; a constant zero point offset and an intensity dependent correction term. The constant offset term can again be separated into three parts; bias level, zero image and dark current image.

The bias level is a positive value that is added to the signal from the output amplifier before sampling, to avoid negative values when the signal fluctuates around zero, since the analog-digital converter cannot handle negative counts. This number is constant over the image, and is usually determined by sampling more pixels from the CCD chip than there are physical pixels (*i.e. overscan*), and taking the average of these.

The zero image is an image of the CCD chip with zero exposure time, after the overscan level is removed, and determines variations in the zero points of individual pixels. It is determined by taking a number of frames with zero exposure time and averaging them. Most modern CCDs have no significant zero level structure. If structure can be seen in the zero image it is most often due to ground currents that will vary from frame to frame, and cannot be corrected for by simple subtraction. If no structure is seen in the zero frames it is safe to leave out this correction term.

Dark current, as described above, must be corrected for if an absolute zero point is required. For cryostatic systems dark current is insignificant in modern CCDs even with exposure times on the order of hours. For Peltier-based systems the dark current must be monitored and corrected for, but the importance is dependent on the chip type and operating temperature. Some chips have large pixel-to-pixel variations in the dark current that vary considerably with temperature. The most important thing is to keep the temperature stable during observations, or else corrections will be impossible. If there is no structure in the dark current at a certain operating temperature, this correction is not very important for time-resolved photometry, since any dark current will be removed together with the sky level. However, if structure is seen in the dark current, the level for each pixel must be determined at the same operating temperature as for the observations, and appropriately subtracted.

The intensity dependent correction term is known as the *flat-field,* since it is determined by observing an empty patch on the twilight sky or a flat part of the telescope dome. The image obtained in such a way should in principle be flat or uniform, but in practice is not. Such flat-field images will reveal all surface defects on the CCD chip itself, as well as dust or specs on the CCD camera window or on the filters.[1] Dividing a science frame with an appropriately averaged and scaled flat-field image will in most cases correct for such sensitivity variations.

[1] As Craig Mackay put it: "The only uniform CCD is a dead CCD." (In: *Ann. Rew. Astron.-Astrophys.* vol. 24, p. 255 (1986).

1.3.4 CCD observations and noise

Noise in CCD observations is a lot more complicated than for photoelectric detectors. Of course, all the problems associated with atmospheric transparency will be the same, although in a CCD frame sky background information is abundant, allowing more sophisticated schemes for sky level correction. The design of the CCD instrument is such that all the individual pixels can be sampled by the same electronics, although several amplifier chains may be available. This means that any gain drift in the amplifier electronics during the night will affect all data in the same way, eliminating much of the $1/f$ noise problems often seen with multi-channel photometers.

The limiting noise factor in CCD systems is the readout noise, which depends on the quality of the output amplifier on the CCD chip, when the sampling electronics have been well designed. Typical CCD systems today achieve readout noise levels between 5 and 10 electrons per pixel sampled. Whenever observations are readout noise limited the noise in the observations would be

$$\sigma^2 = p^2 + R^2, \tag{1.8}$$

where p is the photon noise in the signal and R is the readout noise, and all numbers are in photon terms. Note that the data units that the cameras sampler circuits produces are in analog/digital units (ADU) and must be multiplied by the conversion factor of the sampler electronics; the *gain* factor, g, to get proper photon statistics. For the CCD cameras considered here the gain factor is close to unity, and can thus be ignored.

For CCD observations to be readout noise limited it is necessary to take great care in the preprocessing of the data. Every processing step on the raw data will introduce a *reduction noise* term into the data, so it is important to take a significant number of calibration frames and average them together so that the noise introduced by each correction step becomes as small as possible. We will not go into the details of the readout noise terms in this work, since they have been elaborated on many times before. A good review is given by Howell [How92].

For aperture photometry a useful version of the CCD noise equation (*e.g.* [ODo95]) can be written as

$$\sigma = 2.5 \ \log\left(\frac{1+\sqrt{N+n_{\rm pix}R^2}}{N}\right) \tag{1.9}$$

where N is the number of photons detected, R is the readout noise and $n_{\rm pix}$ is the number of pixels containing the star image. Here the sky level is assumed to be zero and the system is assumed to be limited by readout-noise.

1.4 CCDs for Time Resolved Photometry

So far, real time-resolved photometry has mostly been the realm of photomultipliers alone. When this project was started, a number of experiments with CCDs had been made, but no efficient system were available at any major observatory that we were aware of. An early CCD imaging photometer with special capability for occultation observations were described by Dunham *et al.* [Dun85]. They used two small CCDs to observe the field in two bands simultaneously, and implemented windowing to reduce the amount of data. Although this is the first system that actually implemented a windowing solution, the idea had already been suggested earlier by several authors [Dun85].

The first system that were designed with high-speed photometry particularly in mind was presented by Stover & Allen of Lick Observatory as early as 1987 [Sto87]. Their system was based on a frame-transfer solution, so it did not require windowing. To avoid their computer systems choking on the data, they implemented simple aperture photometry in the CCD controller, so that no images were ever saved when the system operated in continuous mode. Despite the fact that their CCD was a thick EEV chip with no sensitivity below 4500 Å, they achieved very impressive results. Comparing results from their CCD photometer running at the 1 m Lick Observatory with Ed Nather observing with his classical PMT photometer at the 2.1 m McDonald Observatory on three stars with different spectral types, they found that they could match the photon counts for some stars even with a collecting surface area only a quarter of that of the PMT device, although for the very blue objects their results were comparable but not better than for the PMT instrument when correcting for the difference in aperture. Despite these impressive results, CCD systems were not used in photometric campaigns until a decade later.

Many authors have used existing systems for CCD imaging at observatories around the world to produce differential photometry well below one millimagnitude per second (see *e.g.* Kjeldsen & Frandsen [Kje92], Gilliand *et al.* [Gil93]). However, these experiments reveal significant problems. Most notably the observations are hampered by the long readout time between exposures, resulting in quite low duty cycles, especially for the brighter targets. The sampling rate can also be irregular, since most astronomical CCD detector systems systems are designed to take one frame at a time, save the often huge data file to disk storage, display the image and then proceed with the next frame. Since the time the computer spends storing the image is not strictly constant on most multitasking computer systems, the delay between frames can vary.

To achieve accurate photometry with CCDs it is vital to perform precise calibration and reductions of all the individual frames of an observation run. In spite of these problems, very impressive results have been made on time-resolved photometry of open clusters, where CCDs have the great advantage of being able to produce accurate magnitudes of all resolved objects in the field. In particular, a detailed

multi-site study with large (2.3 – 5 m) telescopes were performed by Gilliand et al in 1992 [Gil93], in which 12 stars in the open cluster M67 were monitored for a one week period. Although no pulsations were detected in any of the sample stars, the campaign could quote detection thresholds down to about 20 μmag. However, the effort invested when processing and comparing such huge amount of data with traditional CCD reduction and analysis software, is discouraging, and may be a reason why such extensive CCD campaigns have not been repeated.

For a single target star with variability timescales of minutes, multi-channel photoelectric photometry is still the instrument most widely used when observing pulsating stars. In photometric campaigns like the WET, some observatories have used CCDs since 1998 [ODo00], but not achieving the same temporal resolution as the classical instruments.

Our confidence, since starting this project, has been that using the windowing technique with a well designed CCD camera, we will be able to achieve results that are at least as good as with classical photoelectric photometry, and with with significant advantages during variable sky conditions. This way, we can increase the ammount of useful data in a given campaign. A particular ambition here in Tromsø has been to be able to utilise the long winter nights here in the Auroral zone for long photometry runs at a single site. With our CCD photometry method this can now become possible.

1.4.1 CCDs and the WET

The first experiment with CCD photometry during a WET run were made during the XCOV16 campaign during April/May 1998, at the SAAO 0.75 m telescope (South Africa) and the LNA 0.6 m telescope (Brazil). Further tests have been made at other observatories during XCOV17, XCOV18 and XCOV 19, but classical PMT devices still dominate the picture.[2] A one night test run on PG 1336−018 performed shortly after XCOV17 at the NOT with the system described in this work is presented in chapter 5.

During the XCOV16 campaign the SAAO 1.9 m telescope was observing the same DAV star, BPM 37093, with a PMT instrument, simultaneously with the CCD observation on the 0.75 m telescope. A comparison of the results are give by O'Donoghue *et al.* in the Proceedings of the 5^{th} WET Workshop [ODo00]. From this study they conclude that "CCDs are potentially much superior to PMTs: a CCD on a 0.75 m telescope produced data almost as good as that provided by a PMT on a 1.9 m telescope". Their CCD images were produced by a frame-transfer camera with 10 second exposure time/sampling interval, and the data were processed using four different procedures. When two point spread function (PSF) fitting programs (adapted versions of Daophot and Dophot) were compared with simple aperture

[2] See `http://wet.iitap.iastate.edu` for details.

photometry programs (IRAF/apphot and the authors own photkron), it was made strikingly clear that the aperture photometry procedures performed much better than the more sophisticated PSF photometry programs. They write: "It is surprising that aperture photometry appears to be so much better than the two PSF fitting programs tried out here. It might be expected that a method in which a model is fitted to the data with pixels weighted according to the flux from the star should be superior to a simple addition of pixels in an object aperture. The reasons for the poor performance of the PSF method are not known."

The conclusion by O'Donoghue *et al.* [ODo00], that aperture photometry can be superior to PSF photometry, is in accordance with my own experience from many years of photometric monitoring of gravitational lens systems [OEs96]. PSF fitting procedures can perform well in complex systems where several stars and even galaxies overlap, when aperture photometry fails completely. But for an isolated star image the fitting errors introduced, first when determining profile parameters from a number of reference stars, second when centering and scaling this semi-analytical profile to the target star, can be quite significant. Since this problem was anticipated at the start of this work, no attempt has been made to include PSF photometry into the system presented here, at the current stage. We have seen cases where stars are so close together that PSF photometry could have improved the results, but for most variable stars this is not an issue.

One problem anticipated when applying CCDs to WET type targets are amplitude differences. White dwarf and subdwarf stars are hot objects and pulsations are expected to exist predominantly in the blue end of the spectrum. A study of such amplitude differences based on both synthetic and experimental data were given by Kanaan et al, also for the 5^{th} WET Workshop [Kan00]. Since PMT detectors are only sensitive to light at frequencies higher than 7000 Å, while CCDs have impressive quantum efficiencies long into the near infrared region of the spectrum, pulsations in these stars are expected to appear weaker in CCD data when the modulations are normalised to the brightness of the target star. In this study three CCDs were compared to a standard PMT detector, and the models predicted amplitude differences of 12 – 21 % for DAV stars, 10 – 15 % for DBV stars and very small ($\sim$0.5 %) differences for DOV stars (since their spectra are almost flat in the spectral region that the detectors are sensitive to). When comparing the experimental data from the XCOV16 run on BPM 37093 from the two SAAO observatories, as described above, they found that amplitudes in the PMT data were between 10 and 20 % stronger than in the CCD data, as expected. Their study also showed how this effect could be reduced to around 5 % by using short pass filters like a $CuSO_4$ filter or a wide blue-green BG39 filter.

1.4.2 Other advantages

Some other advantages with CCD detectors should also be mentioned. Many interesting astronomical targets have companion stars, or close foreground or background stars that can appear within the instrument aperture. This can seriously muddle up the observations, if the aperture cannot be made large enough to include all sources. Any star left on the edge of the aperture will wander in and out during observations, and introduce spurious variability in the light curves. The same can be the case for the nuclei stars in bright planetary nebulae (PNNs) or any other star superimposed on a bright non-uniform background. With a set of CCD images, the background object can be easily modeled and subtracted from the image after convolution with the PSF whenever it can be assumed to be constant on the time-scale of the observations. Thus, many new targets become accessible to high-speed photometrical studies.

Another advantage is that since most observatories are equipped with (or are in the process of acquiring) a CCD imaging camera or CCD spectrograph with imaging capabilities, time resolved photometry can be done with existing instruments. Since CCD detectors are quite expensive compared to PMT devices, for most small or medium sized observatories one CCD instrument that performs many tasks is a cost efficient solution. This is an important reason why we have rejected the frame-transfer solution as an acceptable alternative for our CCD photometer. Too many astronomers are concerned about field coverage to accept the introduction of a light-blocking mask covering half of the effective CCD array. By implementing the windowing solution we avoid this conflict of interest, and are able to implement efficient time resolved photometry on a number of existing instruments.

A final point to mention is that the abundance of data existing in a series of CCD images give excellent opportunity to apply more advanced and better reduction techniques to the data, than what can be possible with the simple output of PMT instruments. In particular, the sky background estimates can be significantly improved by increasing the number of pixels to average over, or even introduce surface fitting techniques to model the non-uniform gradients in the bacground often seen when moonlight is scattered inside the telescope structure.

1.5 The Purpose

After this introduction it should be clear that there is no need to prove that CCD instruments can give improved photometry also for time-resolved photometric observations, since this has been done many times over. The urgent need is to make such observations workable at our observatories and within the WET framework. For this we need several things. First of all, we need to establish a CCD observation strategy that can provide the temporal resolution required in observations of short

time periodic variable stars. We have chosen to implement the windowing technique on the CCD controller level instead of employing frame transfer CCD devices, so that we can apply our solution to many existing CCD cameras. At the same time we need to establish the optimal requirement for a CCD traveling photometer to replace the Pancake traveling photometer for WET campaign participation of observatories that does not own their own instrument. For this purpose we have here in Tromsø taken upon us the development of our own portable CCD camera based on the Copenhagen CCD controller. The technical part of this design will be presented in the next chapter, and the modifications of the CCD controller code necessary to implement the windowing mode will be presented in the third.

Once we have established the hardware platform that we will use for our CCD photometer, we will start tackling the considerable problem of CCD data acquisition and processing. The windowing method will produce CCD data that are not reducible with any existing software, so we need to develop everything from scratch. The WET requirement, that light curves must be available in real time for transfer to the coordinating center, demands a system unlike something ever before attempted in CCD photometry. The development of the CCD data acquisition and reduction software will be the subject of chapter 4, while the remaining chapters will discuss the testing of these programs on variable stars of the sdB class.

Chapter 2

The Hardware Guide

Although the Astrophysics Group at the University of Tromsø has no previous experience in CCD camera design, we have good experience in designing and building classical photometers. Our experience should be adequate for building the mechanical parts of the camera as well as the complete filter and shutter unit. Although our previous instruments were not operating in vacuum, we have an excellent plasma laboratory at the department that can provide the equipment and experience we need for this part of the job. When it comes to the CCD sequencing and sampling electronics, we have decided to draw on the experience built up over several years at the Copenhagen University Observatory (CUO) when building high performance CCD camera for, among others, the Nordic Optical Telescope (NOT), and purchase a complete CCD controller electronics system from them, instead of attempting to develop our own system.

2.1 Design limitations

Before we enter the description of the CCD photometer system and the individual hardware components, we will give an outline of the particular considerations that we have put into our design. We can summarise the primary design goals as follows:

1) Optimal photometric accuracy for short (< 1 min) exposures

2) High sensitivity (limiting magnitude >15 with 1-meter telescopes) for all optical bands.

3) Frame rates of 10 seconds or better

4) Lightweight and portable design

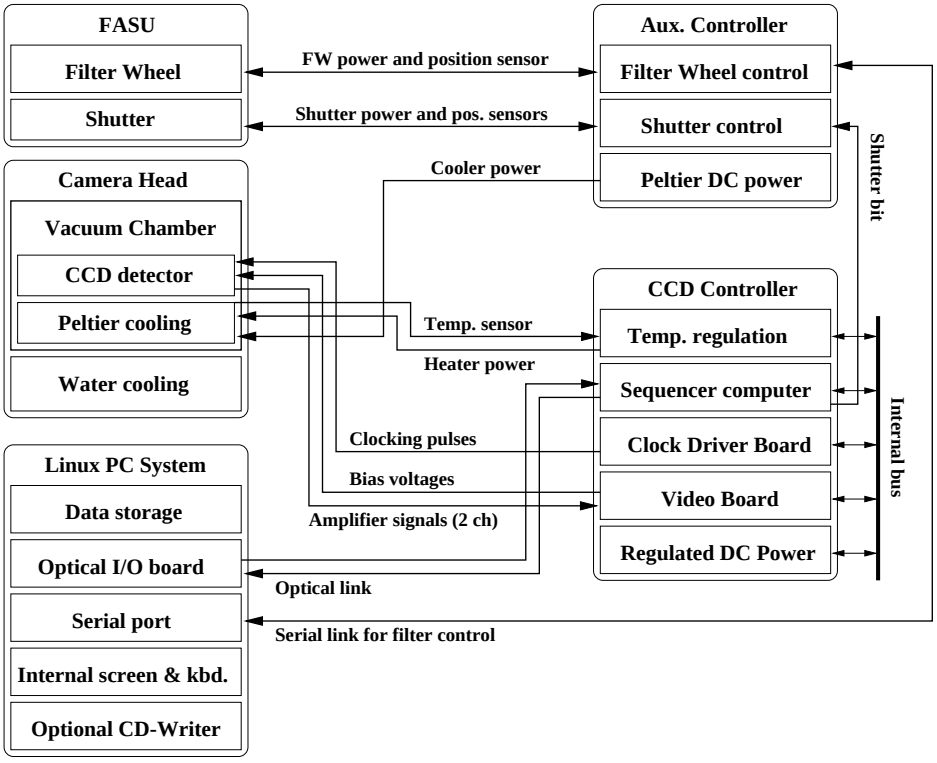


Figure 2.1: Principal design and interactions of the Tromsø CCD photometer. Two controller boxes interface the CCD detector and the filter and shutter unit (FASU). The Linux PC system provides a graphical user interface, data storage and backup systems, as well as network access where available.

The first point reflects our intention to focus on high speed photometry of individual stars, instead of deep imaging of faint objects. Not emphasizing a requirement for long exposures allows us to use thermoelectric instead of cryostatic cooling. The second point introduces the requirement for sensitivity in the blue part of the spectrum, where most pulsating stars have their highest brightness. This limits our choice of CCD chip to the more expensive thinned devices. The third point states our requirement for a new readout strategy to reduce the readout period, *i.e.* windowing. The fourth and last point states our portability requirement, since we wish to use the instrument as a traveling photometer during WET campaigns, in much the same way as the Pancake photometer has been used before [Mei93b].

With such a compact design strategy, we have made the complete CCD photometer fit into three hand luggage pieces, the first containing the CCD camera and its controller, the second holding the FASU and its electronics, and the last piece being a portable PC-unit. The whole system with PC and cables, weighs less than 30 kg.

2.1.1 Operating temperature considerations

Since the exposure times used in high-speed photometry are much shorter than what are used for deep CCD imaging, the cooling required for the CCD photometer need not be as deep as for the CCD cameras built by the Copenhagen group. For exposure times of only 10 seconds the dark-current in the CCD will not be a problem unless the temperature gets much too high.

For the TK1024 CCD chip we have dark a dark current estimate of about 1 e^-/hour at $-95°C$, from measurements with StanCam on the NOT. Extrapolating according to the theoretical formula in Eq. 1.7, and using the information given by the manufacturer [Tek91], we get approximately 1 ke^-/hour at $-60°C$, and 100 ke^-/hour at $-25°C$. Thus, even at $-25°C$ we have less than 300 e^- of dark current in a 10 second exposure. This means that between -30 and $-25°C$, dark current becomes a noise source of the same order as the readout noise and sky background under normal conditions. This is acceptable but not optimal, so we will strive to get the cooling efficient enough to keep the operating temperature at about $-40°C$, where we expect a dark current of about 500 e^-/minute. Note that if we use a device with MPP technology, the expected dark current is five times lower than for non-MPP devices such as the one we have based our calculations on here [Tek91].

We conclude that a triple Peltier-junction cooler will be sufficient to reach useful noise values. A Peltier based cooling system will dramatically reduce the weight and size of the instrument. Furthermore, no liquid nitrogen would be needed to be brought to the observatory, as would be a considerable weight problem when using a cryostatic dewar. However, a Peltier system would require water cooling for transporting heat away from the camera, at least under summer temperature operation. This is not an uncommon problem at observatories, especially at Southern latitudes, and most observatories carry adequate solutions. In any case, ordinary tap water would be sufficient to transport away the heat. We can also use a close circuit system with a bucket of ice and water placed outside the dome together with a small electrical pump and some water hose. It is important in this context to realize that it is not sufficient to run the cooling system at maximum to keep the dark current as low as possible in the individual frames. Thermal stability is the most important issue for time resolved photometry, since variations in the dark current would turn up in the observations during the night. Some part of the chip may show more dark current at certain temperatures than others, thus a normal background level correction may not be trusted upon to accurately remove the dark current with the sky level. It is therefore necessary to regulate temperature by monitoring and correcting the temperature as close to the chip as possible. This has been done by attaching a temperature sensor and a heating device to the thermal connector between the Peltier-junctions cold face and the opposite of the illumination surface of the chip (actually the front-side). By measuring the temperature with this sensor, and regulating a the current through the heater at a high frequency, to correct for deviations

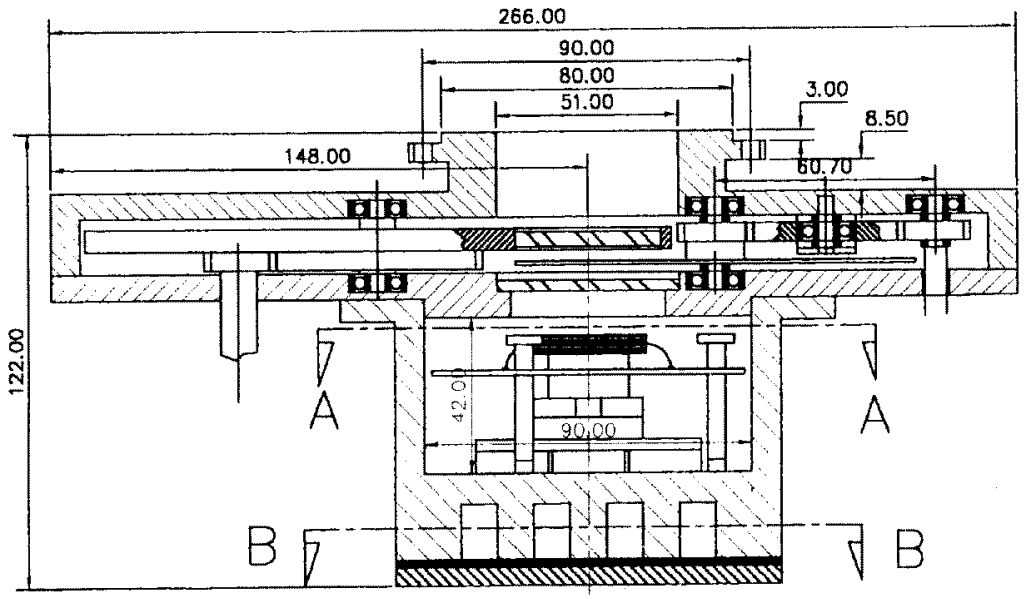


Figure 2.2: A drawing of the first prototype version of the Tromsø CCD photometer, showing a vertical section through the camera head mounted on the filter and shutter unit. Horizontal sections through points A-A and B-B are shown in Fig. 2.4.

from a reference temperature, the temperature of the CCD silicon substrate can be kept very stable, only limited by the precision of the sensor (normally $\pm 1^{\circ}$C).

2.1.2 Chip choice

The acceptable size of the CCD chip is constrained by by the field required to locate usable reference stars for calibration. This is a problem that get worse as our target objects gets brighter due to stellar number statistics, but eventually the field size of the instrument will always limit the possible targets the instrument can do. If the field size gets too large in relation to the number of pixels on the chip, the pixel size will be large compared to the point spread function of the telescope, resulting in poorly sampled stars. A single object should occupy a reasonable number of pixels on the chip, so that it is possible to average out pixel-to-pixel variations in our magnitude measurements. For this reason we must reject chips with very large pixel size as well as very small. Thus, we end up with a choice of chip that is quite similar to what is used in standard issue astronomical CCD Imaging cameras. Thinned Loral 2K chips would be good as gold, but are in short supply and have shown considerable surface defects in many production runs [And98].

Kodak 2K chips have also been considered. They have surface areas that is a bit smaller than the TK1024, but still adequate, and provide excellent sampling of the stellar images. However, these chips are not available in a thinned version,

Table 2.1: Typical specifications for thinned TK1024 CCD chips.

Property	value	unit
Format	1024×1024	pixels
Pixel size	24×24	μm
Imaging area	24.6×24.6	mm
Dark current (MPP)	0.1	nA/cm^2 (20°C)
Dark current (non-MPP)	0.5	nA/cm^2 (20°C)
Readout noise	7	e$^-$
Full well (MPP)	100.000	e$^-$
Full well (non-MPP)	300.000	e$^-$

and since most variable astronomical targets that we want to study with high-speed CCD photometry are very blue objects, the lack of sensitivity below 4000 for the Kodak chips makes them a poor choice. A new version with enhanced blue sensitivity recently introduced on the market improves the usefulness of the Kodak chips considerably, but they still have quantum efficiencies less than half that of thinned chips over most of the visible spectrum. EEV chips have recently become the preferred choice by most professional CCD camera manufacturers, due to their excellent specifications and wide range of sizes and dimensions. They are, however, forbiddingly expensive, and for our prototype we have decided to settle on a more reasonably priced device. Therefore, we have ended up with the choice of the thinned SiTe TK1024 CCD chip for our prototype based on considerations of the chips physical properties; dimensions and sensitivity, chip supply and the cost advantages of using available camera electronics designs. The detailed specifications typical for these chips are given in Table 2.1. The actual properties may differ somwhat from chip to chip.

2.1.3 Camera parts

Our CCD camera can be divided into four separate parts. The *camera head* holds the CCD chip in a vacuum chamber that is in close contact with a cooling system. On one side of the camera head a transparent window lets light into the vacuum chamber and onto the CCD chip. Around this window is a flange that attaches to a unit that holds the filter wheel and shutter – the *filter and shutter unit* or *FASU* for short. The other side of the FASU attaches (through an adaptor ring) the camera to the telescope. Both the camera head and the FASU each have electronics boxes that contains the electronics and power supplies for their individual units, and are separately interfaced with the user computer.

The weights and dimensions of these parts are given in Table 2.2, well within the design goals. As we can see, the dominating part both in weight and size is actually

Table 2.2: CCD photometer dimensions

Part	Weight	Size
Camera head	1.5 kg	$15 \times 15 \times 10$ cm
CCD controller	3.5 kg	$24 \times 24 \times 13$ cm
FASU	2.5 kg	$27 \times 18 \times 3$ cm
FASU controller	3.0 kg	$23 \times 22 \times 11$ cm
User computer	10 kg	$40 \times 29 \times 21$ cm

the user computer. In the following we will describe each of the camera units in more detail.

2.2 Camera Head

The camera head has been designed as a small compact unit with as few components inside the vacuum chamber as possible. It holds the CCD chip, which is thermally connected to the Peltier element by a copper block called the *cold finger*, a molecular trap, and some minor electronic parts. All electrical connections are brought into the vacuum chamber by two vacuum plugs. On the back side of the camera head, the side facing away from the telescope and optics, a cooling system transports away the heat generated by the hot side of the Peltier element.

The window that admits light to the CCD chip is made of 3 mm thick fused silica, and is 40 mm in diameter. It is held in place on top of a groove in the aluminum lid in which a viton o-ring rests. When the vacuum chamber is pumped out the glass is firmly pressed towards the o-ring, and is locked in place by a copper ring from above. Two electrical vacuum connectors and one opening for pumping the camera are also firmly sealed with viton o-rings. The vacuum pump connection is of type KF-16 (16 mm diameter), and the vacuum valve is shown on the left side in Fig. 2.3. Fig. 2.2 incorrectly shows the vacuum chamber in direct contact with the FASU back side. In the actual prototype version there is a lid that closes the vacuum chamber with a 10 cm (external diameter) o-ring between, as seen in Fig. 2.3.

2.2.1 Vacuum chamber

Inside the vacuum chamber, in full contact with the water cooled bottom plate, the Peltier element provides a thermal gradient of up to about 70 K. On top of this the cold finger provides thermal conduction between the cold side of the Peltier element and the CCD chip. The hot and cold surfaces of the Peltier element have both been covered with a special type of thermally conductive high-vacuum grease to

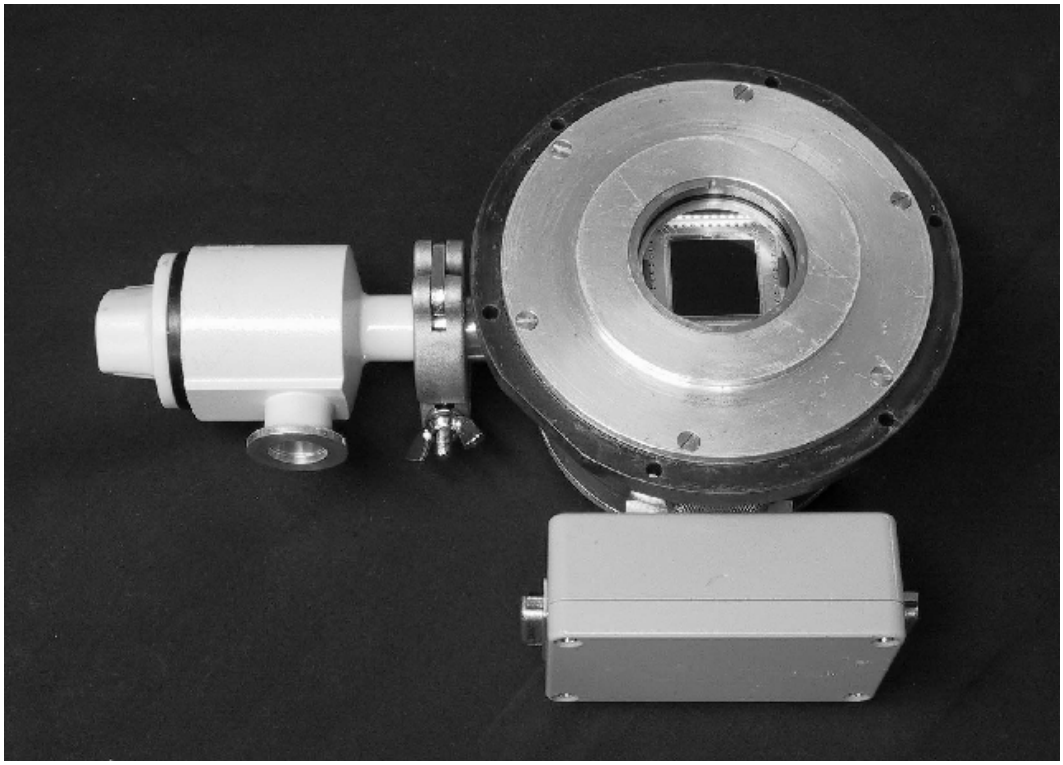


Figure 2.3: The finished camera head prototype with the TK1024A CCD chip installed. The pre-amplifier box is at the bottom and the vacuum valve to the left.

provide optimal thermal conduction. Attached to the cold finger is a temperature sensor (close to chip) and a heater resistor. The most efficient way to keep the temperature as stable as possible has proven to be by running the cooling system on maximum and regulating a small current through the heater to keep the temperature constant at the cold finger, instead of trying to regulate the temperature through the Peltier element. This system is described in detail in the section on the temperature regulation electronics board, below. The cold finger has been drilled through to provide a cylindrical cavity that has been filled with activated coal to work as a *molecular trap* when the chamber has been evacuated.

The Peltier element is a Melcor three stage device, that can draw up to 3.5 Amperes of current at 15.4 Volts. At maximum power it can provide a thermal gradient of 96 K, but in practice it then generates far more heat on the hot side than can be transfered away, so under normal conditions 7 to 8 Volts is the maximum useful power applicable, which will give a thermal gradient of about 50 K. With efficient water cooling some more power can be used to provide a thermal gradient of 70 K. It is crucial that the whole back side of the Peltier element is in full contact with the bottom wall in the vacuum chamber, or else the heat will dissipate back through the device and reduce the thermal gradient. In the current assembling of the camera a piece of soft indium were inserted between the thermal surfaces to further enhance

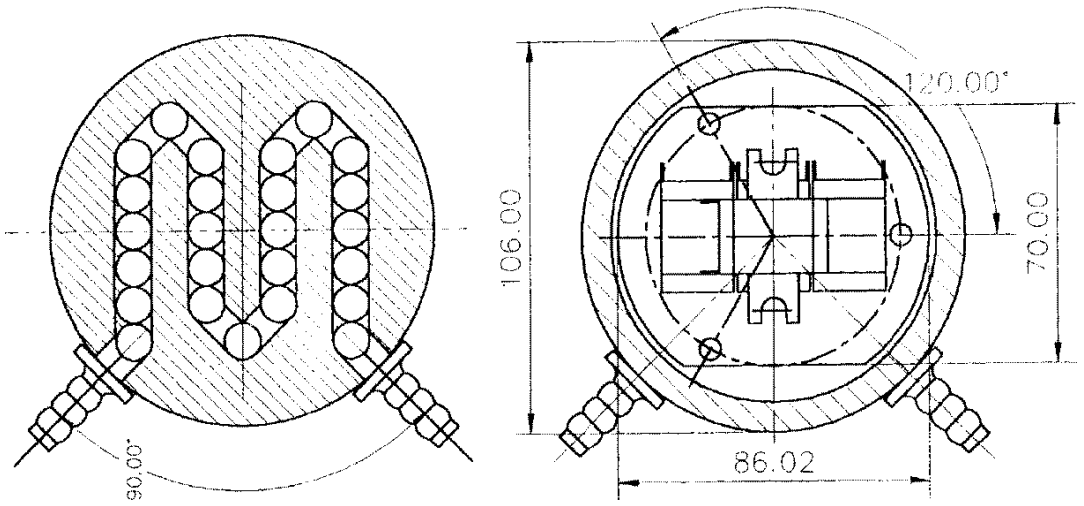


Figure 2.4: Horizontal section through the camera head. Left part shows the water cooling, and the right part shows a view of the inside with the *cold finger* copper block installed (see text).

thermal conductivity.

2.2.2 Water cooling

To provide a stable heat sink for the cooling system, water flows through channels in the bottom of the camera head, as indicated in the left hand part of Fig. 2.4. The temperature of the water should of course be as low as possible, but it is even more vital that the water temperature is kept within well defined ranges. If the water gets to hot, the Peltier cooler will not be able to keep the temperature below the reference temperature, and the regulation system cannot keep up. Too cold is not good either, since it means that a lot of power must be applied to the heater to keep the chip temperature stable. It is vital to know before an observation run starts what the maximum temperature of the water during the observation run can become, and use this value to set the reference temperature on the temperature regulator board.

After testing at Skibotn Observatory (Winter, 2000), and at Moletai Observatory in Lithuania (Spring, 2000), it was clear that the water cooling was not required during the cold winter nights, but was vital for spring and summer operations. In an ongoing modification of the camera head, the internal water cooling part of the camera head will be removed and replaced with an interface upon which at least two different cooling systems can be placed. For wintertime operations a simple heat sink with a fan will be used, while a more elaborate water cooling device will be constructed for summer time operations. Having the water cooling separate from the camera head makes it possible to run and test the water cooling system for any

leaks before attaching it to the camera head.

2.2.3 Connectors

There are a number of connectors to the camera head. Two vacuum connectors provides connection with the electronics inside, a hole for the vacuum pump, and two plugs to attach the water flow. The 16 mm, 26 pin vacuum connector provides power to the Peltier element, and also connects the temperature sensor and heater to the temperature regulation board. The smaller 14 mm, 36 pin vacuum connector interfaces the required pins of the CCD chip directly to the electronics box just outside the camera. This box holds the pre-amplifiers and connects with one RS-232 25 pin cable and two special (BNC + 10 pin) cables to the controller box.

2.2.4 Pre-amplifier box

In the first step in the output signal chain, the output from the CCD amplifiers passes through a pre-amplifier located close to the chip itself. For our system, the pre-amplifiers have been put in a small electronics box attached on one side of the camera head (seen in the bottom part of Fig. 2.3). This is an improvement from earlier Copenhagen designs, in which the pre-amplifiers were located inside the vacuum chamber. This solution will reduce heat dissipation and out-gassing from electronics components inside the vacuum, although the slightly longer wires may increase the noise from the signal amplification. The amplifier design is, however, good enough to ensure the readout noise should be dominated by the noise from the output amplifier itself.

All the voltages from the clock drivers and bias generators are also connected through the new pre-amplifier box, which also holds an improved safe-guard mechanism that protects the chip from any damaging voltages that may occur from static charges or short circuits in the DC connectors.

Although the TK1024 CCD chip has four output transistors (MOSFETs), only two of these are wired to the pre-amplifiers. Since the chips pinning is symmetrical, it can be rotated in the socket to select the two MOSFETs that gives the lowest readout noise. In full frame mode it is possible to reduce readout time by dividing the chip into two or four areas of equal size, and reading these areas through separate output channels simultaneously. In windowed readout mode this would require the windows to be distributed in a symmetrical fashion, which is rather inconvenient. For windowing, we would also prefer that all data have the same noise characteristics, *i.e.* that all pixels are read through the same amplifier channel. Although the system is built with two parallel amplifier channels, we will not give much attention to this, except note that the one with the best noise behavior should always be selected.

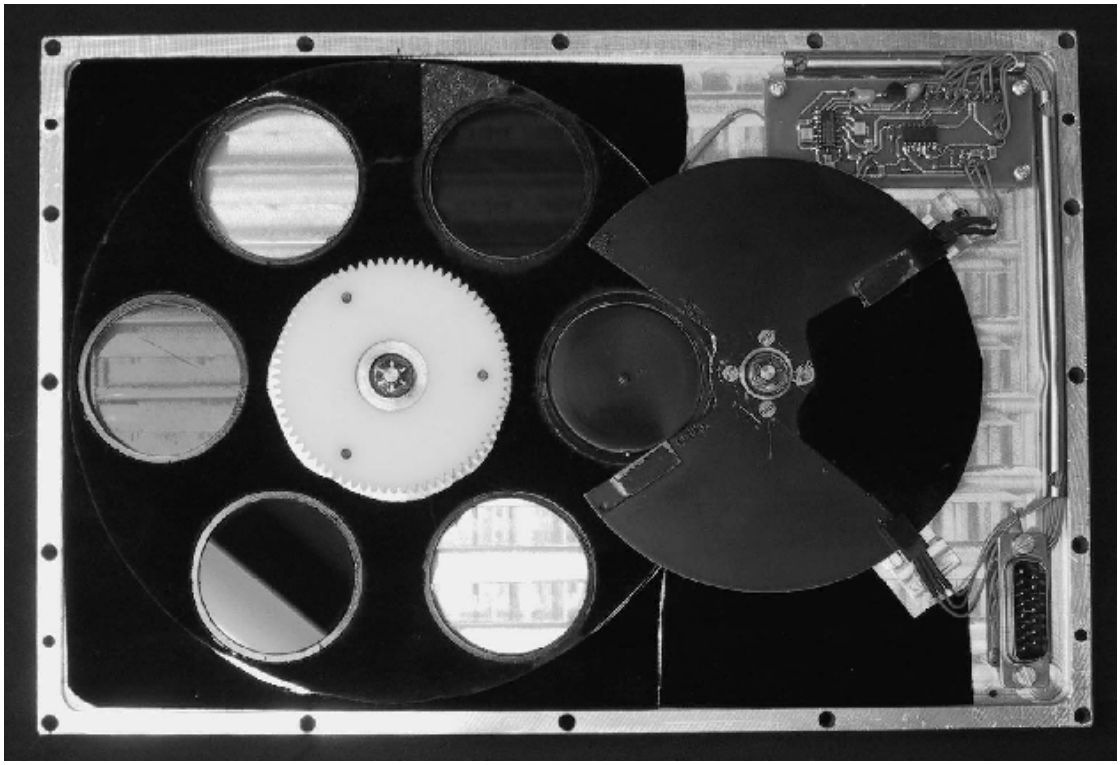


Figure 2.5: Inside look of the prototype filter and shutter unit with four filters installed. Some surface areas have been covered with special light absorbing paper to reduce reflections.

2.3 Filter and Shutter Unit

The Filter and Shutter Unit (FASU) contains the rotating filter wheel with up to six 40 mm diameter round filters, and the shutter wheel. It is necessary for the filter wheel to reproduce its position exactly (to within the size of one resolution element) each time it is turned, or else the flat-fielding of the images will be poor if dust and specs on the filters change position relative to the pixels they obscure. The rotation of the shutter must also be accurate to avoid shutter effects. We have avoided using standard iris shutters, because the operation of these shutters will always give slightly different exposure times across the field, even if they open and close very rapidly. The rotating sector shutter will open for illumination on one side of the CCD chip before the other, but due to its symmetrical design the closing of the shutter, after integration is complete, can completely cancel this difference. To achieve this the acceleration and deceleration of the shutter motor must be symmetrical.

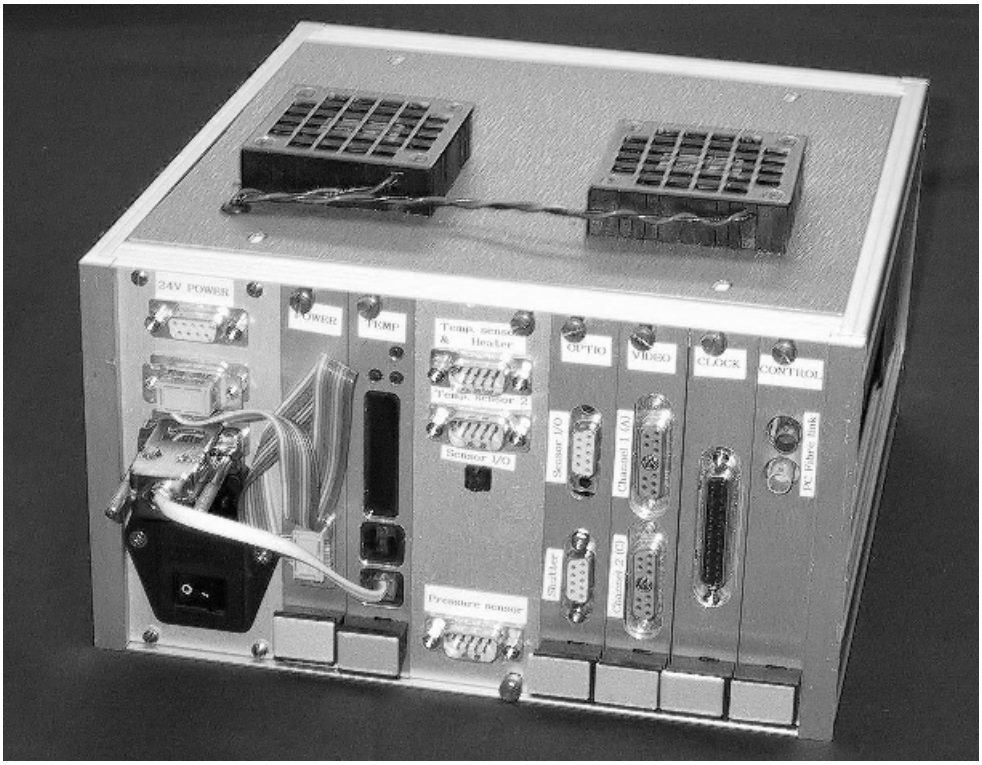


Figure 2.6: The Copenhagen CCD controller.

2.4 The Copenhagen Controller

The CCD controller electronics have been designed and built by the Astronomical Instrument Centre (IJAF) at Copenhagen University Observatory (CUO), and is of the same standard design that are in use on the three CCD cameras operated at the Nordic Optical Telescope (StanCam, HiRAC, ALFOSC), the DFOSC at the Danish 1.5m telescope at La Silla in Chile, the BFOSC at the 1.52m Cassini telescope at Loiano in Italy, and several other observatories around the world [And98].

The Copenhagen CCD controller system has been under constant development for the past ten years by a very small staff of scientific and engineering personnel. While the quality and reliability of their instruments have proved excellent, the documentation provided with their hardware is incomplete. In particular, for a proper understanding of the specifics of CCD readout, that is required to change the operations of the controller to provide multi window readout, it leaves a lot to be desired. The only source of information available on the most detailed levels of the CCD sequencing procedure lies in the original assembly code source programs for the controller that have been provided by Preben Nørregaard of the CUO. Careful examination of these while translating segment by segment into more readable C code, as well as personal discussions with Nørregaard, has uncovered the detailed workings and made possible the coding for windowed mode that is presented in the

next chapter. In the remaining parts of this chapter we will wade a bit more deeply into technical details, of such issues as the specific hardware addressing that modifies the behavior of the individual parts of the controller electronics, than may appear necessary. The reason is simply that this information is not provided in clear text anywhere else, and if somebody at a later time need to modify the code that will be presented in the next chapter, the information given here can work as a useful manual.

The CCD controller electronics is held in a compact aluminum box that must be attached to the telescope in close proximity to the camera head. The limitation is the length of the two cables that connect the output from the pre-amplifiers to the CDS circuits that are 55 cm long. The box contains a power supply and a series of equally sized boards that interfaces through a Motorola backplane bus. There are eight slots for controller boards in the box, but only six are used in our current version of the controller. Fig. 2.6 shows the box, and we see the six cards labeled from right to left CONTROL, CLOCK, VIDEO, OPTIO, TEMP and POWER. The CCD sequencer board (CONTROL) contains the programmable MC 68020 controller computer, the clock driver board (CLOCK) provides the timed signals that holds and shifts the charges on the CCD chip, the video board VIDEO has the CCD sampling circuits and reference voltage supplies, the opto-isolated I/O board (OPTIO) provides optical bridges between noisy external electronics and the sensitive parts associated with the CCD driver and sampler circuits, the temperature regulation board (TEMP) provides the stable temperature for the CCD chip, and the power regulation board (POWER) converts the input power from the 24V power supply to a variety of regulated voltages on the backplane bus. The order of the cards is not vital, but it is customary to put the non noise-critical boards close to the input power supply, and the noise sensitive boards on the opposite side. Between the temperature regulation and the opto-isolated I/O board are the two available board spaces, but as can be seen in Fig. 2.6 the space has been filled with a panel providing the external connectors to the temperature and pressure sensors and the heater, as well as a connection to the OptIO board. All these cards will be described in detail in the following sections.

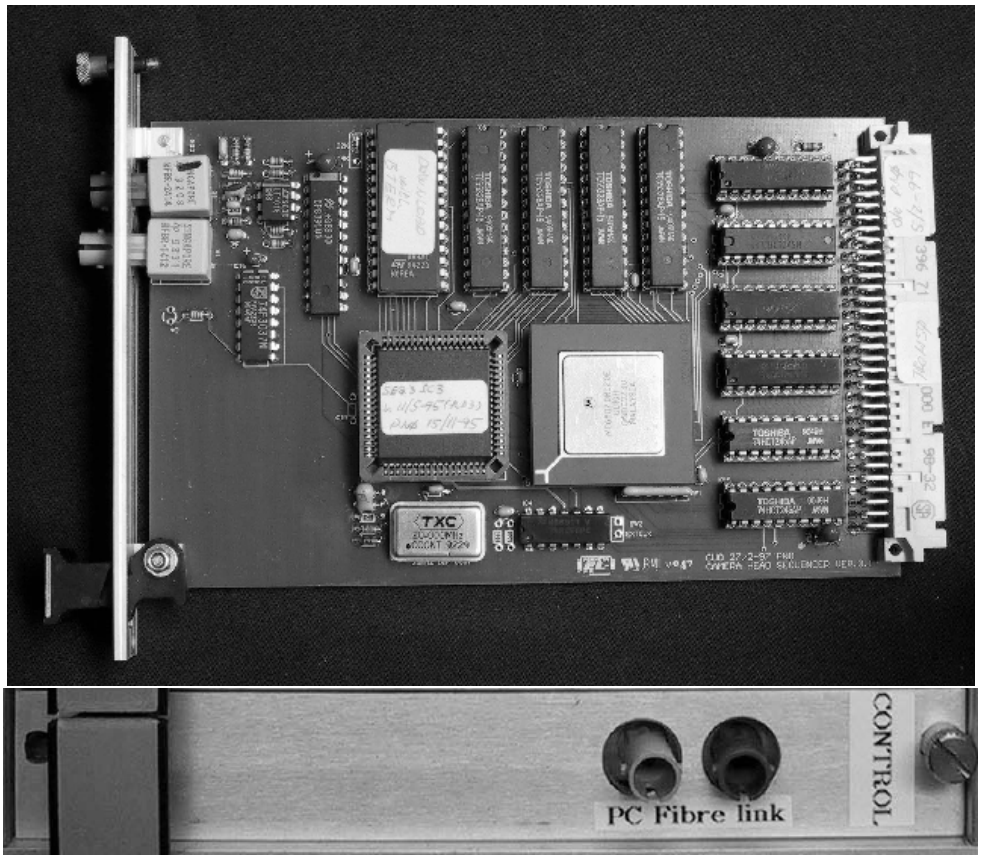


Figure 2.7: The CCD sequencer board with the MC68020 CPU in the middle. Below is the front panel with the fiber optical connectors.

2.4.1 CCD sequencer board

The heart of the CCD controller is the sequencer board. This is in fact a stand-alone computer with a Motorola MC68020 processor running at 20MHz. It is the same CPU that powered most of those millions of classical Macintosh, Commodore Amiga and Hewlett-Packard systems that were popular a decade ago. It is a well documented and programmer friendly microprocessor chip, and with 128 kilobytes of RAM and a 10 megabit optical fibre link, the system is more than powerful enough to handle the data sequencing and transmit the resulting data to the user computer.

On power-up the system starts running from the on-board EP-ROM memory. Normal operation would include copying the main program stored in this permanent memory and over to the system RAM address space, to speed up execution. It is also possible to use an EP-ROM device with a *download* code that allows the user to transmit a program from the user computer directly to the RAM, and then start execution of this new program. This option is very useful for development and debugging, but in the final stage a new EP-ROM with the upgraded version should be

Table 2.3: Sequencer board internal I/O registers

Name	Address	Size	Description
RXTXS	0x40000	byte	Optical I/O register
SETREG	0x40002	word	Status register
DELREG	0x40004	word	Delay trap register
TIMREG	0x40006	12 bit	Timer register
RXDATA	0x40008	word	Receiver data register
TXDATA0	0x40008	word	Primary transmitter data register
TXDATA1	0x4000A	word	Second transmitter data register
TXDATA2	0x4000C	word	Third transmitter data register
TXDATA3	0x4000E	word	Fourth transmitter data register

made and installed.

The system RAM starts at hardware address 0x20000, and two sets of programmable hardware registers are defined to control the hardware circuits on the sequencer board and on the other boards present on the common backplane bus. The internal hardware addresses, INTIO for short, are defined as starting at absolute address 0x40000, and the addresses external to the sequencer, SEXTIO, starts at 0x60000.

A list of the specific internal registers are given in Table 2.3, and those registers are used for the following: The RXTXS register is used to signal activity on the fiber optical transmitter/receiver. The SETREG status register contains a number of important bits that controls what register (or registers) the transmitter should send from, and whether the timer is running or not. The DELREG delay trap register is provided for accurate timing control. Whenever this register is set to a non-zero value, attempting to read it halts the controller until the value is zero. The timer circuit will automatically decrement the value on the register every 20 MHz clock cycle. Thus, setting DELREG to a value of 20000 and then performing a number of instructions before accessing the register, ensures that the time taken from the register was set until the program continues is exactly 1 ms. TIMREG is a timer register that is automatically incremented every millisecond whenever the timer is running, and can be used to measure time between control loops and the length of the CCD readout process. Unfortunately, in the present version of the controller, this register is only 12 bit, which means that it overflows at 4096 ms. Since readout times in general are higher than this, it is necessary to read TIMREG several times during readout (*e.g.* every line), and add up to get the total readout time. This introduces truncation errors that are cumulative. RXDATA is the register where the receiver places the data transmitted to the system on the fiber link, and TXDATA are the four registers where the transmitter fetches data to be sent on the fiber link. The first four bits of the SETREG register specified which (one or many) of the TXDATA registers should be transmitted, allowing up to four words to be transmitted

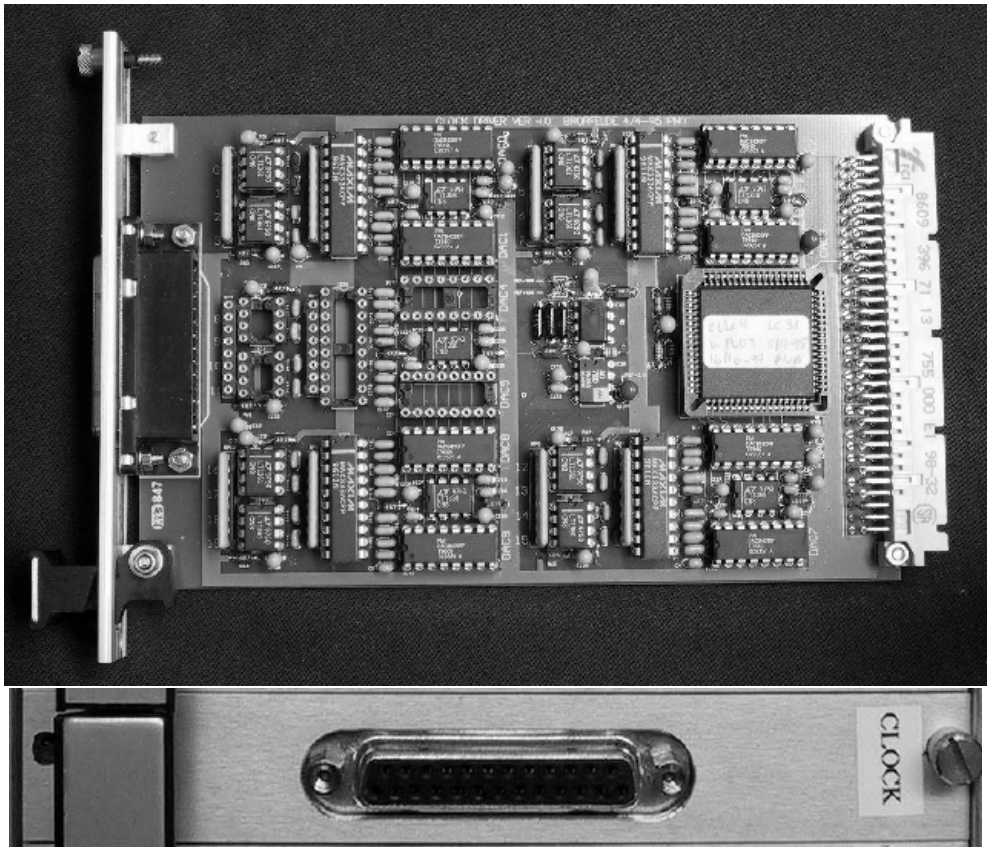


Figure 2.8: The clock driver board has space for 5 identical DACs providing four analog voltages each, but only four are installed on this card.

without the controller having to interfere. This is required when more than one sampler circuit is active.

2.4.2 Clock driver board

The clock driver board provides 20 individual drivers that can generate signals for controlling the charge shifting on the CCD chip. Not all of these 20 drivers are required for a given CCD system. To read out the TK1024 using one amplifier we would require nine drivers: One for the reset gate (RG), one for the summing well (SW), three for the serial shift phases (S1, S2 and S3), three for the parallel shift phases (P1, P2 and P3) and one for the transfer gate (TG). In our system we designate two more parallel shift phase drivers to support dual mode readout, so that we have five parallel drivers (P1U, P1L, P2, P3U and P3L). There is no point in designating an extra driver for the P2 phase since only the P1 and P3 phases have separate electrical connections (pins) for the upper/lower parts of the CCD array on the TK1024 chip.

Table 2.4: Clock driver board registers

Name	Address	Size	Description
COMSCLK	0x60000	word	Serial clock pattern
COMPCLK	0x60002	word	Parallel clock pattern
CLKPT	0x60080	longword	Clock connection register
CLKDAC	0x60084	one bit	Clock driver voltages data

Table 2.5: Bit assignments for COMSCLK and COMPCLK

Bit	0	1	2	3	4
COMSCLK	RG	SW	S1	S2	S3
COMPCLK	P1	P2	P3	TG	

The actual signal that is produced by these drivers is decided by an on-board logic (PLD). It is programmed by feeding pairs of signal and driver identifiers to the hardware address given by the hardware register CLKPT (see Table 2.4). The signal identifiers and their corresponding bits are given in Table 2.6, and the designated drivers as configured for the TCS as discussed above is listed in Table 2.7. Normally the reset gate signal, RG_SIG, would be sent on the reset gate driver, RG_DRV and so on for all the other drivers. The exception is for the five parallel drivers, where mixing of these allows the readout direction to be controlled. Setting the two P1 drivers to produce the P1 signal and the P3 drivers to produce the P3 signal gives output through the A amplifier chain. Switching these two so that the P1 drivers produce the P3 signal and the P3 drivers produce the P1 signal reverses the direction of charge transfer, and thus the pixels can be read out through the B amplifier. To get dual mode readout, that is half the image through A and the other half through B, the upper phases must work as in A mode and the lower phases in B mode; *i.e.* P1 signal on P1U_DRV and P3L_DRV and P3 signal on P3U_DRV and P1L_DRV.

For the Loral 2k chip the situation is different, since it has its two output amplifier on the same serial register. Thus, there is need for only one set of parallel drivers, while the serial register is divided in two halves and the pixels can be clocked left or right, or one half each way.[1]

The clock driver board has a number of on-board registers which can be accessed from the sequencer computer at the addresses given in Table 2.4. The COMSCLK and COMPCLK gives a bit pattern that tells the clock driver board which drivers should be in its high and which drivers in the low state. The CLKPT and CLKDAC registers

[1]We sometimes discuss the Loral 2k chip, even if we have chosen the TK1024 chip for our system. This is due to the fact that existing systems at the NOT use this chip, and we intend to install our multi-windowing system in these cameras.

Table 2.6: Bit assignments for individual clock drivers

Bit	Signal	Description
0	RG_SIG	Reset Gate signal
1	SW_SIG	Summing Well signal
2	S1_SIG	Serial phase 1 signal
3	S2_SIG	Serial phase 2 signal
4	S3_SIG	Serial phase 3 signal
7	SER_ST	Serial stuck signal
8	P1_SIG	Parallel phase 1 signal
9	P2_SIG	Parallel phase 2 signal
10	P3_SIG	Parallel phase 3 signal
11	TG_SIG	Transfer Gate signal
14	PAR_ST	Parallel stuck signal

are used to program the chip specific voltages into the clock driver chip. The clock driver board has several identical DAC chips which each provides four drivers. The CLKPT register is used to select which chip should receive the data, and the voltages is transmitted on the most significant bit of the CLKDAC register. Voltages must be given in millivolt on version 4 of the clock driver board, not in 1/100 volts as on earlier versions.

In Table 2.8 the voltages that the different clock drivers should produce to run the TK1024 are given, together with their allowed ranges [Tek91]. All values are given with a *Low Rail* for using when the specific clock is low and a *High Rail* for the clock's high state.

During integration all drivers are in the low phase except the P2 driver which is set high to produce a barrier that confines the electric charges to each pixel element. During readout the drivers are cyclically activated to shift the potential wells that holds the electric charges stored during integration towards the output amplifiers, with sufficient delays between each shift to allow the electrons to migrate with the moving wells.

The pulse diagram for the parallel shift sequence shown in the left part of Fig. 2.9 shows that how one CCD line can be shifted to the serial register by switching the appropriate clock drivers through seven steps. P2 is the collecting phase, *i.e.* the voltage is held high to collect the photo-electrons at that gate during exposure and while the serial register is read. The first step of the shift cycle is setting the adjacent P1 gate high, to allow the electrons to drift toward that gate. Next the P2 voltage is set to low to get all electrons to collect at the P1 gate. This continues for the P3 and transfer gate (TG). During these seven steps the collecting phase of the serial register (S3) is held high, so that when the TG voltage is lowered in the last step of the parallel sequence, all charge in the line adjacent to the transfer gate has been

Table 2.7: Clock driver assignments for a CCD system with TK1024 chip

Drv#	Name	Description
0	RG_DRV	Reset Gate driver
4	S1_DRV	Serial phase 1 driver
5	S2_DRV	Serial phase 2 driver
6	S3_DRV	Serial phase 3 driver
7	SW_DRV	Summing Well driver
12	TGL_DRV	Transfer Gate driver (lower half)
13	P3L_DRV	Parallel phase 3 driver (lower half)
14	P1L_DRV	Parallel phase 1 driver (lower half)
15	P2_DRV	Parallel phase 2 driver
16	P1U_DRV	Parallel phase 1 driver (upper half)
17	P3U_DRV	Parallel phase 3 driver (upper half)
18	TGU_DRV	Transfer Gate driver (upper half)

Table 2.8: Clock voltages for TK1024 chip

Name	Low Rail		High Rail		
	Range	Set	Range	Set	Description
RGx	–5, 5	0.0	5, 20	11.0	Reset Gate x=a,b,c,d
SWx	–10, 0	–2.0	5, 20	8.9	Summing Well x=a,b,c,d
Snx	–10, 0	–3.0	5, 20	8.9	Serial Gate n=1,2,3 x=a,b,c,d
Pny	–10, 0	–8.0	0, 15	3.0	Parallel Gate n=1,2,3 y=UL,UR,LL,LR
TGy	–10, 0	–6.0	0, 15	4.0	Transfer Gate y=UL,UR,LL,LR

shifted onto the serial register.

When one parallel shift sequence is complete, the serial register is loaded and a sequence of serial shift sequences can begin – one for each pixel on the serial register. The serial shift sequence is very similar to the parallel shift sequence, except for the brief reset gate pulse at the beginning that clears the charge on the output gate. After the end of the seventh step in the serial shift sequence, the pixel charge is on the output gate ready for conversion.

The value set in the two registers COMSCLK and COMPCLK define the state of the serial and parallel drivers respectively. A value of zero sets all drivers to the low state and setting the bits of these registers raises the state of each driver as given by Table 2.5. Thus, setting the COMPCLK register to 1 is all that is required to initialise the controller to collecting parallel phase (P2 high, all other low), for a non-MPP device. In MPP mode, the collecting phase is inverted, so that all voltages are low during charge collection.

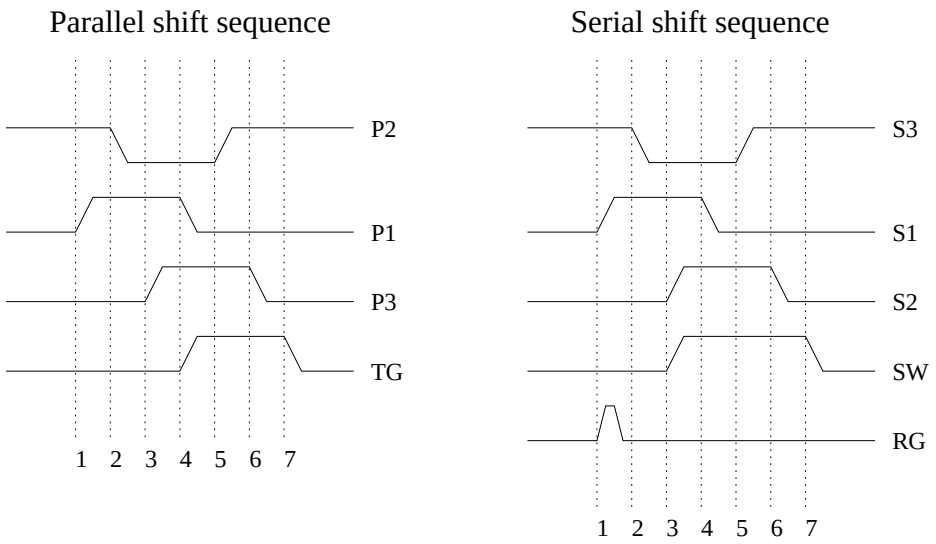


Figure 2.9: Clock pulse diagrams for parallel and serial shift sequences for the TK1024 chip in single amplifier non-MPP output mode.

To check that the clock driver operates as it should it is simple to measure the voltages on the output RS-232 port, using the information in Table 2.9. The table lists the correct voltages that should be found on the 25 pins of the Clock driver board output socket, and the pulse width during CCD clocking operations. Pins numbered 1, 13, 14, and 25 are labeled on the boards socket. During standby or integration mode, all voltages should be low (except P2 in non-MPP mode). All values are for the TK1024 setup. By initialising readout without the camera head attached, it is also possible to use an oscilloscope to monitor that the pulses have the correct widths. Fig. 2.9 shows the pulse diagrams used when moving and sampling charges on the CCD.

Table 2.9: Pinning configuration and voltages for TK1024 clock driver configuration

Pin	Drv	Signal	Low	High	Width	Description
1	-	GND	0.00	0.00	-	Ground, also on 5, 9, 13 & 14
2	19	TGU	-6.00	4.00	80	Transfer gate, upper half
3	17	P3U	-8.00	3.00	80	Parallel phase 3, upper half
4	15	P2	-8.00	3.00	80	Parallel phase 2
6	12	TGL	-6.00	4.00	80	Transfer gate, lower half
7	10	-	-	-	-	Unused driver
8	8	-	-	-	-	Unused driver
10	5	S2	-3.00	8.90	0.4	Serial phase 2
11	3	-	-	-	-	Unused driver
12	1	-	-	-	-	Unused driver
15	18	-	-	-	-	Unused driver
16	16	P1U	-8.00	3.00	80	Parallel phase 1, upper half
17	14	P1L	-8.00	3.00	80	Parallel phase 1, lower half
18	13	P3L	-8.00	3.00	80	Parallel phase 3, lower half
19	11	-	-	-	-	Unused driver
20	9	-	-	-	-	Unused driver
21	7	SW	-2.00	8.90	0.4	Summing well
22	6	S3	-3.00	8.90	0.4	Serial phase 3
23	4	S1	-3.00	8.90	0.4	Serial phase 1
24	2	SWB	-2.00	8.90	0.4	Summing well bias
25	0	RG	0.00	11.00	0.2	Reset gate

2.4.3 Video board

The new video board has integrated the old CDS board and bias generator board into one, and is seen in Fig. 2.10. It consists of two parallel CDS sampling circuits and 3 DAC devices capable of generating 10 stable voltages, known as the *bias generator*. For the TK1024 chip only three DA generators are used for each amplifier chain, providing the *reset drain* (RD), *output drain* (OD) and *output gate* (OG) voltages. The bias generator device is controlled by the same PLD logic and hardware addresses as the video board, and is the voltages are set in software in much the same way as for the clock driver board.

The new bias generator has 12 bit DA converters instead of 8 bit, as in the old version, which gives better temperature stability. The bias level values can also be adjusted in software, whereas the old version required manual trimming with a screwdriver of the potentiometers on the board.

To program the bias level voltages the hardware address of the CDS board in question is used, 0x60020 in our single board system. The DA converter channel is selected by giving a number on bit 1,2 and 3 of this address. Since there are three

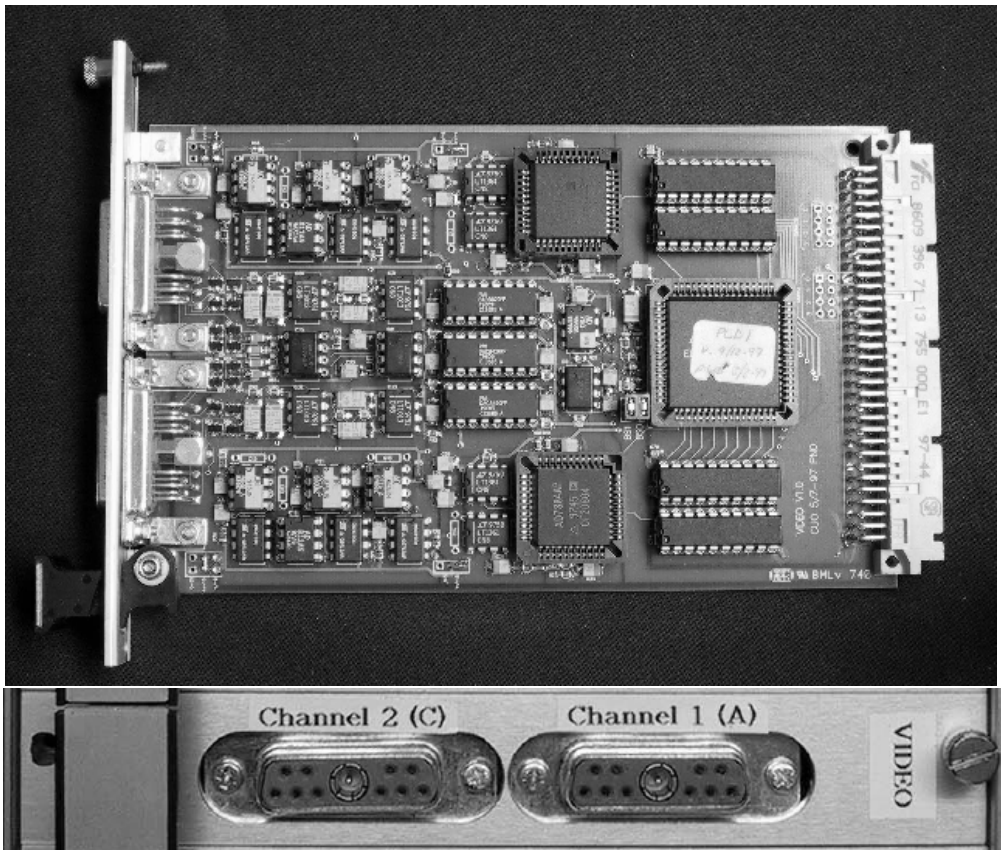


Figure 2.10: The video board has two identical CDS sampling circuits (upper and lower part) and a DAC circuit for the stable bias voltages. The front panel has two identical connectors, one for each amplifier channel.

DA chips (DAC's) on each CDS board providing four voltages each, the mapping from logical to physical channels is non-trivial for some of the output voltages. For the primary amplifier chain, setting the three bit channel selector to 0 selects the OD voltage, 1 selects RD and 4 selects OG. Auxiliary voltages (unused in our system) OGx and ODx are selected by setting the bit field to 5 and 2 respectively. Here, the output voltages 0,1 and 2 is on DAC#1 and 4 and 5 on DAC#2 For the secondary amplifier chain, the OD, RG and ODx voltages are again set by selecting 0, 1 and 2, which are generated by the third DA chip on the board. But since there is no fourth DA chip available, the OG and OGx voltages are generated by DAC#2. Therefore, when these voltages are to be selected we must request channel 6 and 7 on the primary amplifier chain.

After the proper channel has been selected, the voltage data can be transmitted one bit at a time on the `BIAS_DATA` bit. The bit is loaded by flipping the `BIAS_CLK` bit briefly on and off. After all 16 bits have been transmitted, the chip can be activated by flipping the `BIAS_LD` bit.

Table 2.10: CDS board registers

Name	Address	Size	Description
CDSOP	0x60004	word	CDS operation control
CDSMODE	0x60006	word	CDS mode setup
CDSBASE	0x60020	8 words	CDS I/O base address
INDCDSx	CDSBASE+2*x	8 words	8 individual CDS I/O ports

Table 2.11: Bit assignments for DAC registers

Bit	Signal	Description
0	SC_ACTIVE	Signal chain active=1
1	BIAS_DAC0	DAC select bit 0
2	BIAS_DAC1	DAC select bit 1
3	BIAS_DAC2	DAC select bit 2
4	BIAS_CLK	DAC clock state
5	BIAS_DATA	DAC data state
6	BIAS_LD	DAC load state
7	BIAS_CLR	DAC clear state

For the TK1024 CCD chip the bias level voltages have been configured as shown in Table 2.12. The table also shows the allowed voltages ranges for the chip [Tek91], as well as the pins on the video connector socket to which the DACs are connected. All the pins are numbered on the socket.

The actual sampling of the charges that are shifted onto the output gates of the CCD chip are performed by the CDS circuitry on the video board. The board measures and digitizes the signals from the output amplifiers using the CDS process described in Chapter 1. As mentioned, this technique is crucial to eliminate the reset noise in the output signal, that would otherwise completely dominate the total noise in the signal.

The readout noise in the output MOSFETs of the TK1024 chip should be less than 10 electrons, according to the specifications [Tek91]. The new pre-amplifier design, introduced in the latest version of the Copenhagen CCD electronics that we have installed in our system, has third order filters installed to reduce high-frequency noise from the CCD.

The CDS sampling circuits can operate with two different gain modes, known as *high gain* and *low gain*. In high gain mode the conversion factor should be close to 1 electron per AD unit, and in low gain mode the factor can be between 2 and 3 e^-/ADU.

Table 2.12: DC voltages for TK1024 chip

Name	Range	Set	Pin	Description
VDDx	15,25	23.0	1	Output Drain
VODx	10,16	13.0	2	Reset Drain
VLGx	–5,5	0.5	9	Output Gate (Last Gate)
SUB	–10,10	0.0	7	Substrate and Package Ground

Table 2.13: CDSOP register bit assignments

Bit	Flags	Description
0	`CLAMP`	Control of clamp switch: 1=clamp, 0=sample
1	`SAMPLE`	Control of sample switch: 1=track, 0=hold
2	`SC`	Converter control: 1=start, 0=stop

Table 2.14: CDSMODE register bit assignments

Bit	Function	Description
0	Sampler control	clear: Microprocessor, set: High speed logic
1	Gain mode	clear: High gain, set: Low gain

It is vital that the pixel readout rate is highly uniform, because of the settling time of electronic components in the amplifier chain. Temporal variations can transform into variations in the voltages of the amplifiers, which cannot be canceled by the CDS procedure, and therefore will show up as increased noise in the sampled data [McL89, p. 169].

The CDS board has two registers that controls the operations of the system. The `CDSOP` register is used to select the operation mode of the converter, by selecting bit-flags from Table 2.13. For instance, during pixel readout the `CDSOP` register is set to `CLAMP` to signal beginning of the clamp/sample CDS readout cycle. After the clamp timing period is completed `CDSOP` is set to zero to end the clamp signal conversion. Then the charge on the summing well is dumped to the output gate, and the `CDSOP` register is set to `SAMPLE` to signal the sample part of the CDS cycle. When the sample period is complete the charge difference can be converted by setting the SC bit of the `CDSOP` register to signal start of conversion. After completion of the conversion phase the sampled voltage difference is prepared for retrieval in one of the converter output registers `INDCDS`0–7 at the addresses given in Table 2.10.

An advanced CCD controller unit can have up to four video boards sampling up to 8 CCD amplifier chains, as required *e.g.* for the CCD mosaic camera soon to be

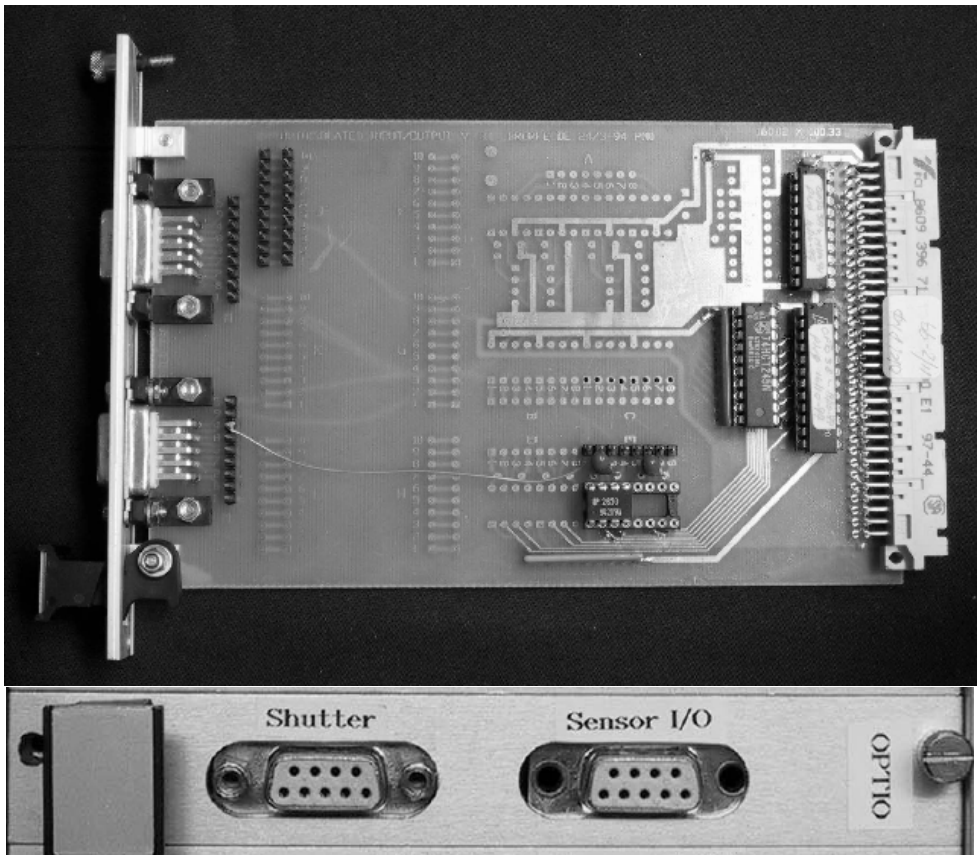


Figure 2.11: The Optio board contains only two single bit opto-isolated bridges. Redundant circuits from earlier controller versions, where temperature regulation were done by the sequencer, have been removed. The front panel provides the connection to the shutter and input from the temperature regulation board.

commissioned at the NOT. Our system has only one video board, and thus only `INDCDS0` and `INDCDS1` are available.

The CDS board used in earlier versions of CCD systems built at CUO were also designed to measure temperature by sampling the voltage drop over a temperature sensing resistor in the system. This option is only used in systems that do not have a dedicated temperature regulation board, such as StanCam (see below).

2.4.4 Opto-isolated I/O board

The *Optio* board provides opto-isolated communications to devices that may have noisy electronics and ensures that the vital sampling circuits remain protected from power surges created when starting and stopping power hungry devices like the shutter and heater unit.

Table 2.15: Opto-isolated I/O board registers

Name	Address	Size	Description
`OPTOBIT`	0x60040	4 bits	Motor and shutter control bits
`OPTOIN`	0x60050	1 bit	Opto input bit
`OPTOOUT`	0x60050	4 bits	Opto output bits
`OPTODEV(0)`	0x60050	word	QDAC (CCD Heater)
`OPTODEV(1)`	0x60052	word	shutter
`OPTODEV(2)`	0x60054	word	step motor #1
`OPTODEV(3)`	0x60056	word	step motor #2

In early versions of the CUO controller design, the sequencer computer also controlled step motors for the filter wheel and shutter as well as temperature regulation of the CCD. This is problematic since the sequencer board is 100% occupied during CCD readout, which means that during readout the CCD temperature will drift and it will be impossible to move the filter wheel. To keep the duty cycle as high as possible it is vital for time series photometry to utilise the readout period for changing filters. In our system, the filter wheel is controlled by a completely separate step motor controller, and temperature regulation is provided by a separate temperature regulation board (see below). The Optio board is now used only for two purposes; to send a signal to the shutter controller when it should open or close the shutter, and to read back sensor data from the temperature regulation board for book-keeping.

The Optio board has been allocated hardware addresses in the range 0x60040 to 0x6007F, and the address names that occur in the code are listed in Table 2.15. The macro `OPTODEV(driver)` writes to the hardware addresses 0x60050–56 `(OPTOOUT)`. There is one word for each driver, but only the least significant bit is used for the opto-I/O conversion. In versions of the controller where the sequencer board regulates temperature and controls the step motors for the filter wheel and shutter, four drivers are used. In our system, only the shutter and heater are controlled through the Optio board. On startup `OPTODEV(1)` is set to activate the shutter, and the macro `SET_SHUTTER(CLOSED)` sets the appropriate bit in `OPTOBIT` to indicate a closed shutter. The `OPTOIN` bit is used in `readtemp()` to transfer the state of the temperature registers on the Tempreg board.

Note that even if `OPTOIN` and `OPTOOUT` has the same value, this does not mean that input and output happens on the same address. `OPTODEV(0)`, which is the heater is used only for reading, and `OPTODEV(1)` which is the shutter is used only for writing. These definitions are more or less obsolete after removal of step motor control from this board, and are kept only for code compatibility.

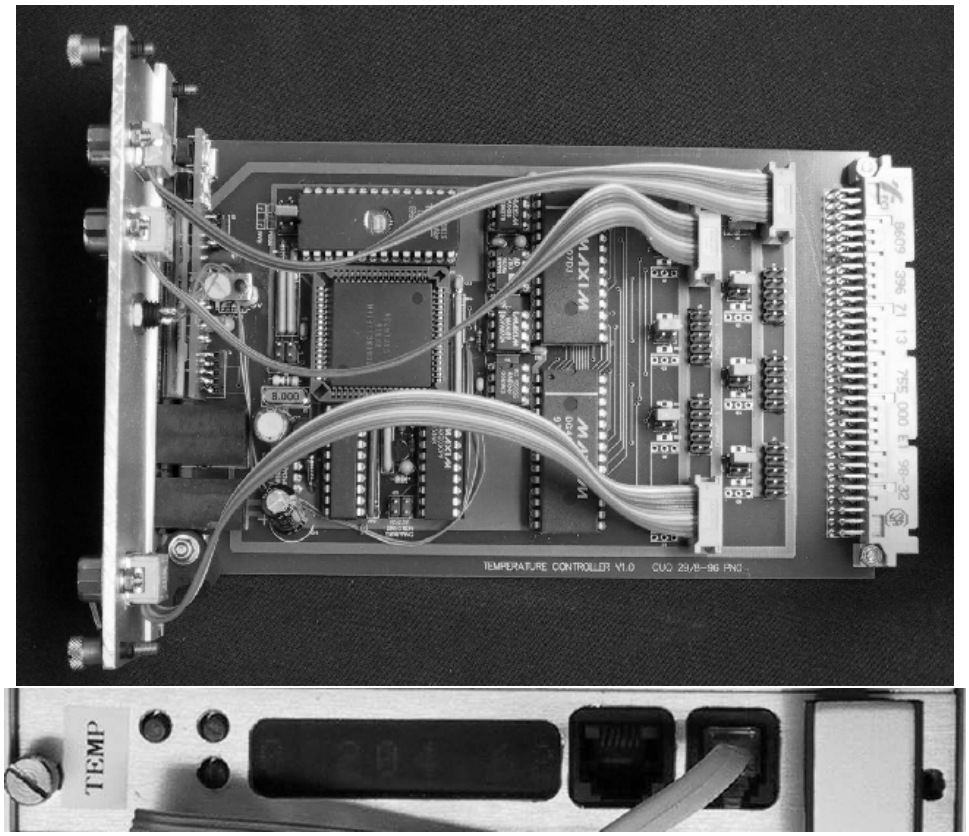


Figure 2.12: The temperature regulation board with sensor connections. The front panel has a LED display as well as three buttons to control and to set reference temperature.

2.4.5 Temperature regulation board

The temperature regulation board is an independent control system based on a programmable logic device (PLD), and provides continuous temperature regulation for the CCD cooling system. In the original version of the CCD control system used for the standby CCD camera at NOT, temperature regulation was made by the sequencer board. This works reasonably well, but no regulation can be done during readout so the temperature may drift significantly during this period. If a new exposure starts immediately after a readout, the CCD temperature can vary during integration. In all new versions of the Copenhagen CCD controller system the separate Tempreg board is present.

The Tempreg board transmits the temperature values through the one bit opto-IO interface provided on the Optio board. The four 16-bit words are continuously transmitted by the Tempreg board, but since the transmission process through the Optio board is very slow and can be interrupted by a CCD readout that takes priority, lost bits can sometimes give bad values.

Table 2.16: Temperature regulation board transmitted data

Name	Size	Description
`ccd_temp`	word	CCD temperature sensor value (regulated)
`ln2_temp`	word	Extra temperature sensor value
`ref_temp`	word	Reference temperature
`pressure`	word	Pressure sensor value

The values collected by the sequencer board are transmitted to the user computer with the rest of the camera status information, as listed in Table 2.16, below. The first value is the measured CCD temperature, which the Tempreg board tries to keep as close to the reference temperature as possible. It can also read a second temperature sensor, which is used for monitoring purposes only. This temperature is the second value transmitted by the Tempreg board. In the liquid nitrogen dewars this sensor is placed in the LN_2 container, and when this temperature starts to rise it can serve as a warning for the user that the dewar is running low on liquid nitrogen and needs to be refilled. For our system we use this extra sensor to monitor the temperature of the water- or fan-cooled backside of the camera. The third value transmitted is the constant reference temperature, which can only be changed through the front panel controls on the Tempreg board. The fourth and final value transmitted is the value of the pressure sensor.

The lower part of Fig. 2.12 shows the front panel of the Tempreg board. The three buttons on the left side of the display are used to control the display and set the reference temperature. The two buttons close to the display cycles up and down through the list of measured temperatures and the pressure. There is some redundancy here, as is also reflected in the space available for connectors seen in the image of the board itself, Fig. 2.12. Space is kept for seven temperatures plus pressure, a total of eight values, but only the two first temperatures and the pressure is relevant for our system. Clicking the leftmost button will reveal a list of reference temperatures, of which only the first `REF0` is used. Clicking the same button again will reveal its value, which is –100.00 by default. The two other buttons will change this number up or down, and the regulation will immediately follow this new value. Searching past the list of reference temperatures will reveal optional display functions, such as displaying the values in ADU or Ohm instead of degrees Celsius. Clicking the leftmost button again when the display reads `INTENS` will turn off the display.

2.4.6 Power Supply

The CCD controller electronics contains a 24V power supply providing 2.5 A for the system. Most of the power goes into the power regulator board (labeled POWER

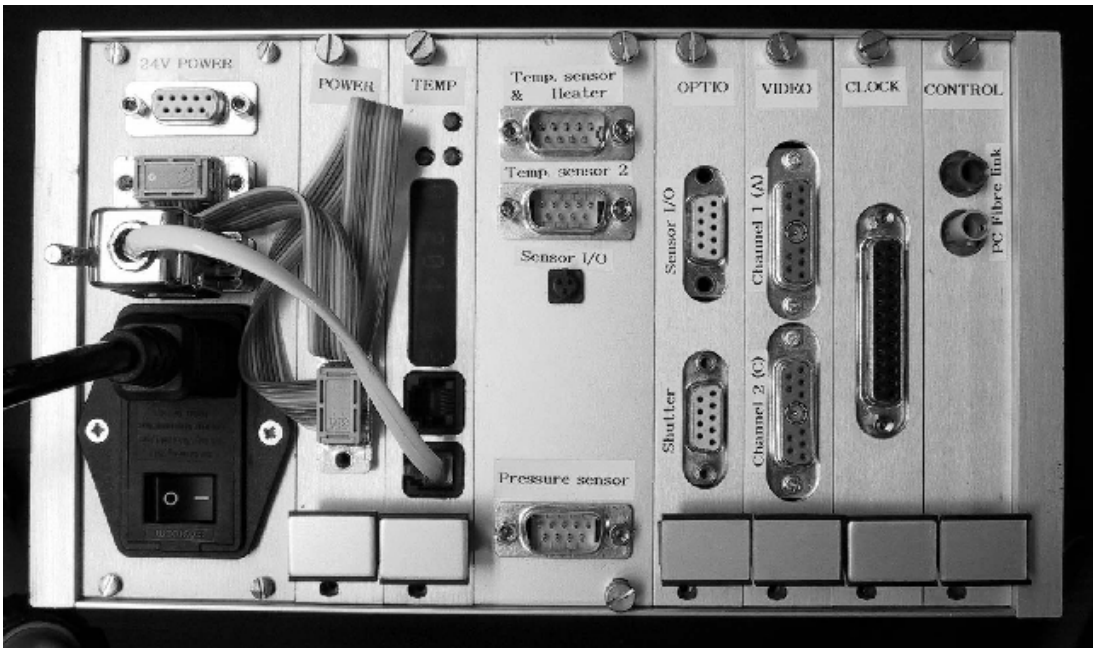


Figure 2.13: The front of the CCD controller box with power cables attached.

in Fig. 2.13) that powers the sequencer computer, clock driver, video and Optio boards. Additional unregulated 24V power goes to the temperature regulation board (through the white telephone cable visible in Fig. 2.13) and to the two chassis fans on top of the controller box (visible in Fig. 2.6). A third 9-pin D-Sub connector with 24V power is available, and can be used to power an additional fan for cooling of the camera head when water cooling is not required (the connector is seen in the upper left corner of Fig. 2.13). The current system draws about 1.5 A of power in normal operation.

2.5 The FASU controller

The FASU controller box contains the two identical API DM-224*i* controllers used to run the filter wheel and shutter step motors respectively. It also holds two power supplies, one that gives 12V power to the two API boxes and one that provides power to the Peltier cooler. These units are marked with arrows in Fig. 2.15. The step motor controllers from American Precision Industries (API) are commercial electronics with well documented interface specifications, so there is no need to go into much technical detail about these. The controllers are fully programmable and contains flash memory that can keep a stored program even when the power is turned off. The command set interpreted by the microprocessors in these units are particularly designed for step motor control, and is known as the IDRIVE protocol. The full specification of this protocol is available from API [API97].

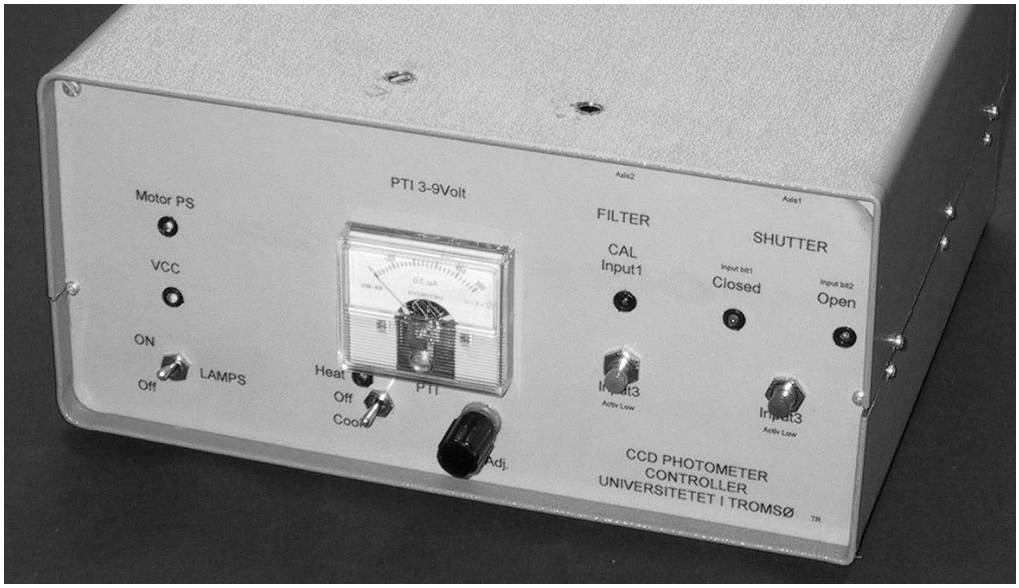


Figure 2.14: FASU controller front side view.

The step motor controllers are programmed and operated from the same user computer system that controls the CCD camera, through an RS-232 three-wire cable. The API boxes can be connected in series (up to 32 units), by giving each unit a unique *axis ID* number selected on five switches located on the inside of the boxes. The shutter controller has been set to axis ID of 1, and the filter wheel controller has been given an ID of 2.

In addition to the step motor interface, each unit takes up to seven input bits and four output bits that can be used to interface the environment. For the shutter controller system, two position sensors, one control button, and one external input connector has been wired to the unit's control interface. The position sensors are magnetic switches that give a signal when the shutter is exactly in open (bit 1) and closed (bit 2) positions respectively. The third bit connects to the shutter test button on the front panel of the FASU controller box. This can be seen on the right side of the front panel shown in Fig. 2.14. The lamps above the test button are hardwired to the position sensors indicating open and closed states, and does not use the API controller's output bits. The fourth input control bit is wired through the 25-pin RS-232 interface to the FASU unit and to the 9 pin input connector on the side of the FASU. This again connects to the Copenhagen CCD controllers opto-I/O interface, to provide accurate shutter timing at start and end of integration.

For the filter wheel controller, only one position sensor is installed and wired to input bit 1 on the respective API controller box, as well as to the lamp on the front panel of the FASU controller box (see Fig. 2.14). The filter wheel test button is wired to input bit 3. No other external inputs are necessary, since the filter wheel is controlled manually by receiving positioning commands directly from the user

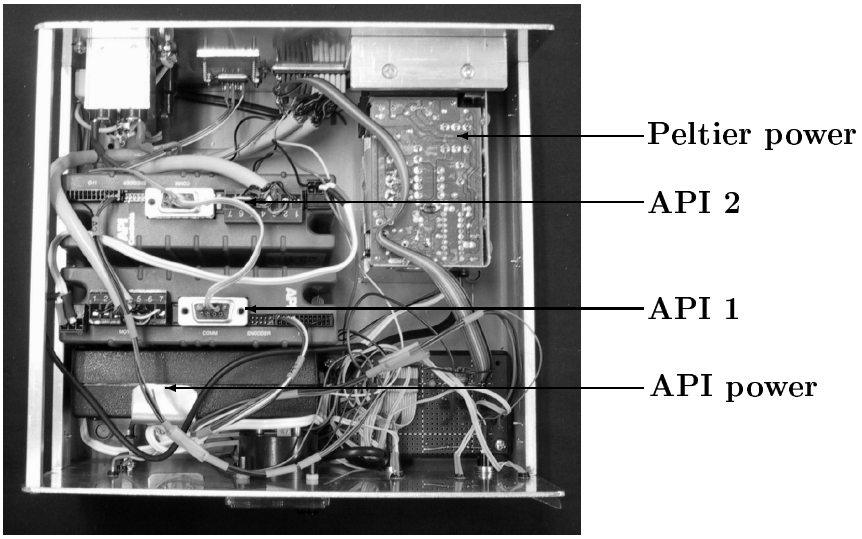


Figure 2.15: FASU controller inside view.

computer.

Besides the control lamps and test buttons, the front panel of the FASU controller unit contains the power control to the Peltier cooling system. The power switch below the voltmeter turns the power on and off, and can also be used to reverse the voltage by holding it in the up position. This is useful to temporarily heat the CCD during pumping to hasten the out-gassing process, but great care should be taken to avoid overheating that can damage the CCD. As a safety precaution the switch will not stay in the up position by itself. Power is controlled with the adjustment knob and the actual voltage applied can be monitored on the voltmeter. The scale (at least in version shown in Fig. 2.14) erroneously reads D.C.μA, but the actual values are volts where a dial reading of 300 corresponds to 6.0 Volt.

On the left side of the front panel are power lamps for the two power supplies, and a button that can be used to turn off all lamps to reduce illumination during observation conditions.

The back side of the box holds the obvious input power connector, the main power switch, a fuse, the output power for the Peltier cooler, and connectors to the FASU and user computer (as shown in Fig. 2.16).

2.6 User Computer System

An ordinary PC with space for an optical fibre I/O board for communication with the CCD controller, and a spare serial port for communications with the FASU controller is required. A compact or portable PC can be used, provided that it has

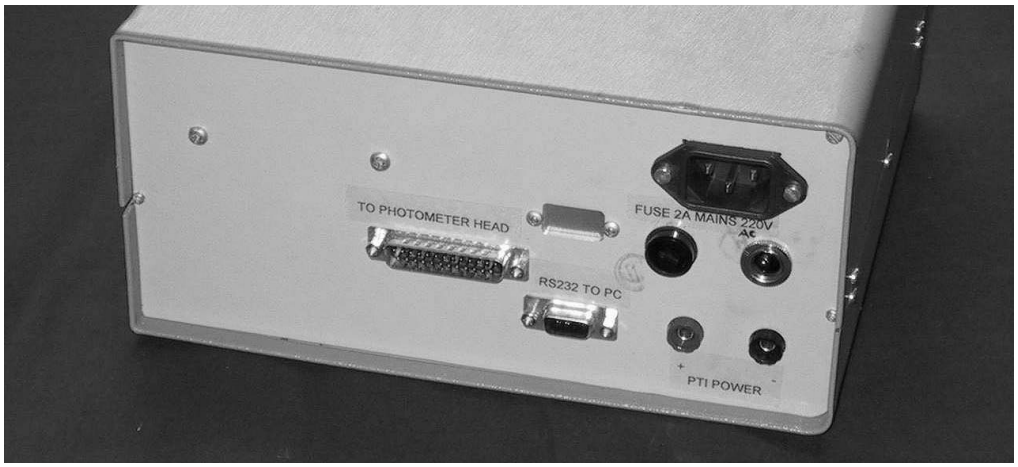


Figure 2.16: FASU controller back side view.

sufficient space for the special ISA bus optical interface card, and a well sized hard drive for storing CCD data.

Linux has been chosen as operating system, by the Copenhagen group, for the user computer system, and we have no reasons to prefer any other system. Real-time operation is only required by the sequencer computer, so Linux is well suited considering its reliable user environment and network accessibility.

For our system we have purchased a compact portable system of the "Lunch-box" type. It is bigger than a standard notebook PC, to provide space for standard type interface cards, as well as standard disks and a CD-ROM reader or writer. However, like good notebooks, it contains a built in high-resolution liquid crystal display screen, as well as a compact keyboard with track-pad, which reduces the size and weight compared to a normal PC system drastically. The system is easy to transport, and has a processing power that is more than well suited for our purposes.

2.6.1 Optical interface board

A significant effort has been put into the optical interface board by the Copenhagen group, to assure that no data is lost as the camera sequencer transmits pixel readout values. Since the camera should sample the pixels at an even rate, normal XON/XOFF handshaking during communications has been abandoned. The interface board receive all data sent from the camera and stores them in a special on-board buffer memory. This memory consists of up to 128Mb of memory in classical SIMM modules, and must come in a group of four identical modules. In the board's logic the receiver has the highest priority, so that no data can be lost. Requests to transmit to the camera and read the buffer memory are handled between receive operations.

Table 2.17: PC-board registers

Name	Offset **Hex**	Size **bytes**	Description
`rwd`	0x00	2	port for read/write to PCBoard RAM
`pcpt`	0x02	4	read/write address
`serpt`	0x06	2	receiver preset register
`txdata`	0x08	2	send data
`txcom`	0x0A	2	send command
`incset`	0x0C	2	autoincrement on R/W [on/off]
`txe`	0x0E	2	transmitter/receiver [on/off]

The optical interface board, or PC-board, works on the privileged system registers from hardware address 0x500 and upwards. There can be up to three different PC-boards in one computer, if they are configured with different hardware addresses. Currently, address bases of 0x500, 0x520 and 0x540 are allowed. Table 2.17 lists the hardware registers used for the PC-board, where the offsets listed are relative to the base address configured for the board. In the next chapter we will describe an interface library that uses these registers to retrieve the CCD image data from the camera.

2.7 Ready to Go

After assembling the hardware components described in this chapter into a working CCD instrument, we would very much like to go observing. However, this type of hardware does not operate just by virtue of their internal electronics. A considerable effort must be put into the software that controls the different parts of the hardware and transmits useful data back to the user computer. This will be the subject of the next chapter.

Chapter 3

Camera Control

Linux was chosen as the preferred operating system for our camera development project at a very early stage. This was done in spite of the fact that the existing development system for the CCD controller was entirely on the MS-DOS platform, when I first started working with the controller code at CUO in Brorfelde in 1994. The motivation was first of all to avoid investing in expensive software, when the GNU compilers and development tools provided a free solution of excellent quality. Another important motivation was that the system requires a reliable platform to run on, and this can only be provided by a system where all source code is available so that any instabilities can be investigated and eliminated. Only the Linux system provides such a solution, and the explosive rise in popularity of this system the last five years, especially in the scientific community, has proven the validity of this choice.

Two independent control systems exists in our CCD camera: The CCD controller and the FASU controller. This chapter describes the software solutions that have been implemented to run on these controllers.

3.1 FASU Logic

The filter and shutter unit is controlled by two API step motor controllers that contain their own programmable logic circuits based on the IDrive protocol specification [API97]. As described in the previous section, these controllers communicate with the user system computer through a serial interface cable, and operate independently of the CCD controller. A library has been implemented in C++ to set up and run the FASU controllers under Linux, since the software provided with the system only runs under Windows.

3.1.1 The FASU Library

The most important commands of the IDrive protocol has been implemented in the C++ library contained in the files `IDrive.h` and `IDrive.C`. This defines a simple interface to send and receive commands on the communication port defined by `COMPORT` in `IDrive.h`.

The FASU class library, in `FASU.h`, uses two instances of the `IDrive` class to implement useful commands like `setFilter()` and `setShutter()` to position the filter wheel or shutter, as well as `getFilter()` and `getShutter()` to retrieve the current positions.

Vitally, the FASU library defines the control sequences that are uploaded to the flash memory of the API step motor controllers. These two programs are named `downloadShutterCode()` and `downloadFilterCode()`, and executes the appropriate `startSequenceStorage()` followed by a sequence of control commands and an `endSequenceStorage()`. The shutter code contains five modules numbered from 0 to 4, and the filter code three modules numbered from 0 to 2.

Shutter Code

For the shutter code, sequence 0 defines the standard initialisation of the shutter controller, it then executes sequence 1 which runs the shutter at slow speed until it locates the calibration position, and jumps to sequence 4, the normal operation mode. Sequence 2 is unused (it seeks the open position at slow speed), but sequence 3 contains a program that is similar to the normal shutter operation program, with the exception that it listens for the shutter open/close signal on the front panel shutter test button instead of the camera shutter signal bit.

The normal shutter control sequence works as follows:

- **Test 1**: If shutter is closed, go to Ready position.
- **Test 2**: If shutter is open, go to Close shutter.
- **Calibrate**: Set target velocity to slow, and rotate continuously until the closed sensor position is found. Set target velocity to high speed.
- **Ready position**: Wait for shutter open signal
- **Open shutter**: Rotate shutter 180° and wait for shutter close signal.
- **Close shutter**: Rotate shutter 180° and go to Test 1.

In sequence 4, this algorithm is preceded by the setup (basically giving the acceleration and peak current, as well as configuring the inputs available and disabling

the de-bounce feature), and a check on the shutter test button. If the shutter test button is pressed at this point (on power up, that is) the program will jump to the test button code in sequence 3. It is also possible to switch between these sequences from the command interface `tcpcom` (see last section of this chapter).

Filter Wheel Code

The filter wheel motor control code consists of three sequences, located in the FASU library function `downloadFilterCode()`. Sequence 0 contains initialisation of the motor parameters, sequence 1 contains the calibration code and sequence 2 a code for the test button. There is no "normal operation code" as for the shutter, since normal operation consists of sending filter positioning commands from the user computer. The calibration code is a simple program that sets the target speed of the filter wheel to 50% of normal speed and moves the wheel continuously until the built in position sensor is triggered, indicating that filter position 0 (usually the B-band filter) has been reached. The test button code is just an endless loop, moving the filter wheel one position forward every time the filter test button is pressed.

3.1.2 FASU Initialisation

The controller code given in the FASU library functions `downloadShutterCode()` and `downloadFilterCode()`, is called only from the program `FASUInit`. Once `FASUInit` has been executed, the code sequences is stored in flash memory in the respective controller units. Normally there should be no reason to re-run this program unless the hardware has been changed. It is, however, a good idea to keep these programs at hand, just in case the flash memory should become corrupted for some reason.

3.1.3 Shutter connection

The shutter is normally controlled by feeding a single bit from the CCD controller to the shutter controller through a separate cable signalling an open or closed shutter state. Note that a high state signals a closed shutter, and a low state gives open. It is possible to control the shutter from the observer PC system, also, but this cannot give the accuracy required for accurate exposure timing, and is therefore only relevant for testing purposes. The commands `getShutter()` and `setShutter()` in the FASU library are used for reading and positioning the shutter, respectively. Note that `setShutter()` will disable any running program if it is called with a open shutter signal, and restart the normal operation sequence (4) if called with a close shutter signal.

3.1.4 Filter control

Filter positioning is controlled from the observer PC by calling the FASU library functions `getFilter()` and `setFilter()`. Since the filter wheel only contains one position sensor signalling that filter 0 is in the current position, the FASU library contains a `filter` variable that keeps track of what the filter position is believed to be. The `getFilter()` function returns the value of `filter`, and makes sure that as long as it is zero the filter wheel is in the calibrate position. If the sensor does not indicate the expected position, the calibration sequence is automatically executed.

`setFilter()` can be called with any valid filter position, and it will compare the requested position with the current value of `filter` to compute the shortest direction and number of steps to move. Unless the filter wheel is already moving, the requested move will be performed. Attempting to move the filter wheel when it is already moving returns an error.

3.2 CCD Sequencing

The CCD controller program that runs on the MC68020 sequencer computer is based on the DFO10 assembly code for the CCD cameras on the DFOSC, ALFOSC and HiRAC built at CUO, and written by Preben Nørregaard. It has been ported from its original MS-DOS environment to Linux, and now compiles with the GNU C compiler, `gcc`, in cross-compiler mode.

3.2.1 Control logic

Once the bootstrap procedure (in `boot.S`) has initialized the computer registers, interrupt vectors and memory, the main program (in `main.c`) starts up by initializing the CCD system parameters by calling the `reset()` function (in `init.c`). The `reset()` function sets all the system variables to correct values for the current system. In particular, it ensures that all voltages to the CCD clock driver board are set up correctly for the specific chip. The `main()` program continues the initialisation by sending a `CLOSED` signal to the shutter and enabling the one millisecond timer interrupt. Then the `main()` program then goes into an eternal loop that checks for exposure timeout, control timeout and receiver interrupts (see flow diagram in Fig. 3.1).

The order of the three tasks in the main loop indicates its priority. Reading out a completed exposure has the highest priority and commences immediately. The control loop is called at a regular interval (set to 35 ms) – except during readout – to send an information header to the user computer, providing a full report of the status of the system. Finally, if there is data on the receiver, the `receive()`

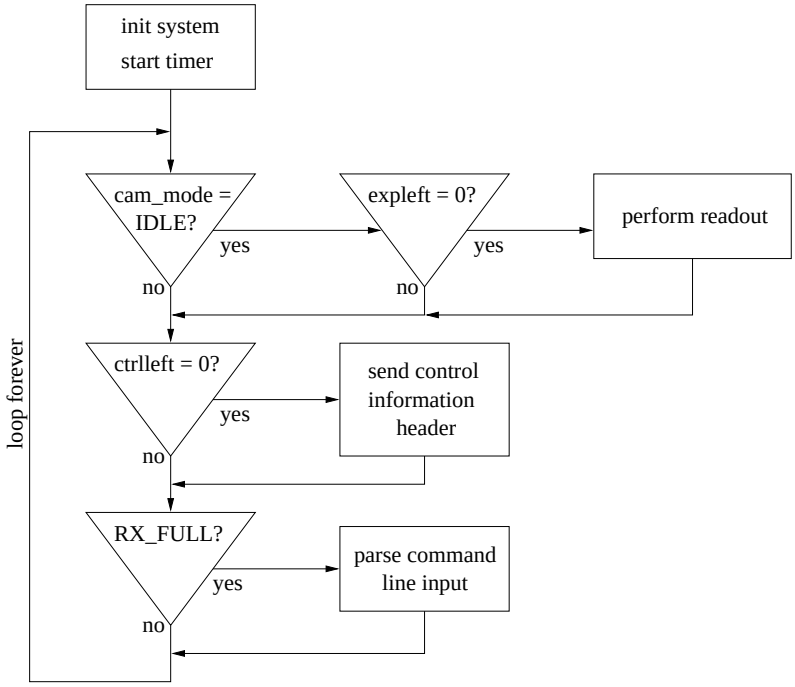


Figure 3.1: Main control loop flow diagram.

procedure in `input.c` is called to read this character and appends it to the input buffer. If the character read is a CR (end of command input signal), the `receive()` procedure will call the `input()` procedure to interpret and execute the command in the buffer.

The main loop is interrupted by the timer every millisecond as set by the timer register `TIMREG` when the timer bit in the setup register `SETREG` is set. The timer interrupt procedure `tim_int()` takes care of the clocking of the control system by incrementing or decrementing the various timing variables. The control time is decremented until it reaches zero, but the actual call to the control procedure `control()` is done in the main loop. The exposure timing variable is also decremented until zero, and again the actual call to the readout procedure is left to the main loop. However, once the expousure time reaches zero, `tim_int()` makes sure that further interrupts are disabled until readout is complete by disabling the timer interrupts. For accuracy, the signal to close the shutter is sent immediately (if the shutter is set to active, *i.e.* `auto_shutter` is 1). A flow diagram of the timer interrupt procedure is shown in Fig. 3.2.

3.2.2 Header initialisation

Almost all variables used in the camera control program are kept in a rigidly defined data structure as given in `cam_info.h`. This status information header is 1024 bytes

Table 3.1: Camera status information header (first 76 bytes)

Name	Offset **Hex**	Size **bytes**	Default	Description
cam_id	0x00	4	"BROR"	Camera header identifier
frame_type	0x04	1	0	Data frame type identifier
cam_mode	0x05	1	IDLE	Camera mode flag
ccd_type	0x06	16	CHIP	CCD chip and code identifier
totx	0x16	2	DEFTOTX	Total # of pixels per line
toty	0x18	2	DEFTOTY	Total number of CCD lines
beginx	0x1A	2	DEFBEGX	Start readout at pixel no.
beginy	0x1C	2	DEFBEGY	Start readout at line no.
imx	0x1E	2	DEFIMX	Number of pixels/line to read
imy	0x20	2	DEFIMY	Number of lines to read
binx	0x22	2	DEFBINX	Serial binning factor
biny	0x24	2	DEFBINX	Parallel binning factor
ccd_temp	0x26	2	-	CCD temperature (measured)
ln2_temp	0x28	2	-	Cold point temperature (measured)
ref_temp	0x2A	2	-	Reference temperature
int_period	0x2C	2	0	Unused in TCP version
readout_time	0x2E	2	0	Actual readout time
frame_left	0x30	2	0	Remaining frames (counter)
frame_total	0x32	2	0	Total number of frames in sequence
cds_setup	0x34	2	(*)	CDS setup (copy of CDSMODE)
mpp_mode	0x36	2	ON	Multi pinned phase [on/off]
auto_clear	0x38	2	ENABLE	Erase CCD before exposure
auto_readout	0x3A	2	ENABLE	Start readout after exposure
auto_shutter	0x3C	2	ENABLE	Active shutter control
active_motor	0x3E	2	-	Unused in TCP version
shutter_delay	0x40	2	SHDELAY	Shutter delay in milliseconds
over_scan	0x42	2	OVERSCAN	Number of overscan pixels/line
exptime	0x44	4	1000	Exposure time in milliseconds
expleft	0x48	4	-	Remaining exposure time (counter)

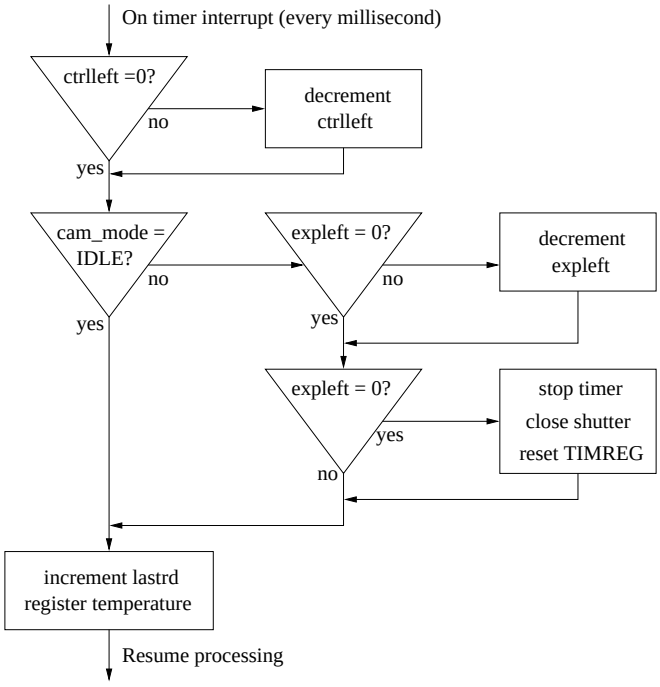


Figure 3.2: Timer interrupt flow diagram.

long, and is transmitted to the user computer system every time the `control()` procedure is called from the main loop (every 35 milliseconds), and at the beginning of each CCD frame readout. The most important header parameters are listed in the two tables 3.1 and 3.2. Initialisation values that are not chip or camera dependent are defined in the file `defs.h`. Note that some of the variables are related to the step motor control and temperature regulation procedures implemented for StanCam, and are obsolete in later versions. The chip dependent parameters are kept in separate files, `tk1024.h` for the TK1024 chip or `loral2k.h` for the Loral 2k chip.

The system allows up to four individual amplifier chains to be active simultaneously, although for our system only two chains have been physically wired from the CCD chip and only two pre-amplifiers are present. These are amplifiers labeled A and C in Fig. 3.3. The readout direction can be changed at any time during camera operation by issuing a command that changes the value of `readdir` to any of the values listed in Fig. 3.3, available on the system in question. The default value is set in `defs.h`.

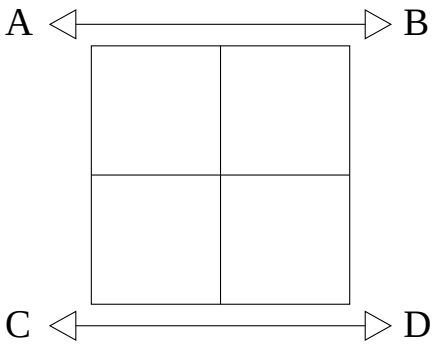
In the current implementation we will only use one amplifier both for normal imaging and window mode operation, although it would be possible to reduce readout time somewhat by using both amplifiers. For normal imaging it is possible to have dual mode readout, provided that the receiving end software is able to handle the output. The option is implemented in the CCD controller, but not in our receiving

Table 3.2: Other important header values

Name	Offset **Hex**	Size **bytes**	Default	Description
`readdir`	0x54	1	C	Chip readout direction
`lastrd`	0x60	4	-	Counts time since last clear [ms]
`tsam`	0x82	2	TSAM	Pixel sample time
`tcla`	0x84	2	TCLA	Pixel clamp time
`tser`	0x86	2	TSER	Serial shift settling time
`tpar`	0x88	2	TPAR	Parallel shift settling time
`sweep`	0x8A	1	TRUE	Clear all unsampled pixels
`ctrltime`	0x8C	4	35	Time between control calls [ms]
`ctrlleft`	0x90	4	35	Counts down time until control [ms]
`seqtime`	0x94	4	-	Time between sequence frames [ms]
`im_tx`	0x98	1	full	Image transmit [None/Full/Partial]
`pressure`	0xAE	2	-	Camera head pressure (measured)
`vclk`	0xC8	80	Table 2.12	Clock driver voltages
`vrd`	0x118	16	Table 2.12	Reset drain voltages
`vog`	0x128	16	Table 2.12	Output gate voltages
`vod`	0x138	16	Table 2.12	Output drain voltages
`checksum`	0x3FC	4	-	Sum of first 74 header bytes

program, `tcpcom`, described at the end of this chapter. Our reason for avoiding the dual readout option is mostly practical. Firstly, to have any gain in readout time it is required that the areas for readout are brought symmetrically to all active amplifiers for sampling. This would make the window setup procedure quite complicated, and possibly introduce noise due to uneven pixel clocking. Secondly, the noise characteristics and conversion factors of the different output amplifiers can be significantly different, and we would have to keep track of different bias levels for the two amplifiers. In the future we should certainly consider how to make best use of this possibility, but for the prototype version of the software we will ignore it.

After the `init()` procedure has initialised all the basic parameters in the header to the startup values (as given in Table 3.1 and Table 3.2), the voltages are initialised. This is accomplished by setting the fields of the header arrays `vclk`, `vrd`, `vod`, `vog`, `vgx` and `vdx` to the desired voltage levels in millivolts. The voltage values are given in the chip dependent setup file, `tk1024.h` for our system. After the arrays have been set up with the correct numbers, `init.c` starts the initialisation procedures for the clock driver board (`clock_init()`, defined in `clkcon.c`) and for the video board (explicitly defined in `init()` to activate the respective drivers). Then the `setdacs()` procedure (in `volts.c`) is called to convert the voltage values from digital to analog values and transmit the values to the relevant drivers that will then start providing the proper output voltages.



Output direction codes:

0: A	4: AB	8: AD
1: B	5: CD	9: BC
2: C	6: AC	10: ABCD
3: D	7: BD	

Figure 3.3: Definition of amplifier output direction codes

After the initialisation is completed the `main()` procedure enters an eternal loop, where it will receive commands to start integrations and readouts from the user computer.

3.2.3 Camera commands

The camera control program reads commands from the user computer system, when it is not busy sending data or sequencing the CCD. When the camera reciever sees the '@' character, the input buffer is reset, and each subsequent character is appended to the buffer until the CR character is received. The first four characters received are interpreted as the command, and the following data is left in the buffer for the specific command to use. The command is processed by the `inline()` procedure which interprets each command by converting any required data values in the input buffer to integer values and setting the proper fields in the `cam` header struct. There is no feedback on illegal commands or variables. Unrecognized four character commands are simply ignored, and error checking on parameter values is quite limited. Either offending values will be ignored, or the proposed value will appear as interpreted by `inline()` in the next header page transmitted.

Check Table 3.3 for details on commands for setting specific header parameters. The command set shown here is compatible with the standard `dfo10` set, except for the two extra commands `@WINS` and `@SEQT` that have been added to support windowing. `@WINS` is used to initialise the list of windows coordinates `wins`, and `@SEQT` can be used to specify the precise time between sequence exposures. As can be seen from Table 3.3, most commands simply sets a header parameter or toggles an on/off flag. A few commands take more specific actions like the `@SINT`, `@FRAM`, `@CLRA`, `@IBRK`, `@FRES` and `@DOWN` commands. These are described in more detail below.

`@SINT` stands for *start integration*, and is the proper command to start a single frame exposure. The command will immediately stop any timer interrupts to avoid the timer to interfere while the CCD chip is beeing erased. If the shutter is enabled, it will be opened. Integration is signalled by setting the `cam_mode` flag, and the

Table 3.3: Set of four character commands recognised by the new control software. N is the number of digits or characters following the command.

Command	N	Sets	Description
@TIME	8	exptime	Set integration time
@SINT	0	cam_mode, expleft	Start integration
@IBRK	0	cam_mode	Integration break
@CLRA	0	-	Clear CCD frame
@READ	1/2	readdir	Toggle/Set readout direction
@TSAM	4	tsam	Set pixel sample time
@TCLA	4	tcla	Set pixel clamp time
@TSER	4	tser	Set serial settling time
@TPAR	4	tpar	Set parallel settling time
@XBEG	4	beginx	Set readout to begin at pixel #
@YBEG	4	beginy	Set readout to begin at line #
@XNUM	4	imx	Set readout number of pixels/line
@YNUM	4	imy	Set readout number of lines
@XBIN	4	binx,imx	Set serial binning factor
@YBIN	4	biny,imy	Set parallel binning factor
@OVER	4	over_scan	Set number of synthetic overscan pixels
@MODE	1	cds_setup, CDSMODE	Set CDS mode
@TMPP	0	mpp_mode, COMPCLK	Toggle MPP mode
@TACL	0	auto_clear	Toggle automatic clear before exposure
@TARD	0	auto_read	Toggle automatic readout after exposure
@TASH	0	auto_shutter	Toggle automatic shutter
@SSHT	4	shutter_delay	Set shutter delay in milliseconds
@VCLK	7/8	vclk[x]	Set clock driver voltage
@VBRD	7/8	vrd[x]	Set reset drain voltage
@VBOG	7/8	vog[x]	Set output gate voltage
@VBOD	7/8	vod[x]	Set output drain voltage
@FRAM	4	frame_left	Start integration of # frames
@FRES	0	All format parameters	Format reset
@SWEP	1	sweep	Clearing of unsampled pixels [On/Off]
@IMTX	1	im_tx	Set image transfer to none/full/windowed
@WINS	18	wins	Initialize windows array
@SEQT	8	seqtime	Set time between sequence exposures
@DOWN	0	-	Jump to download procedure

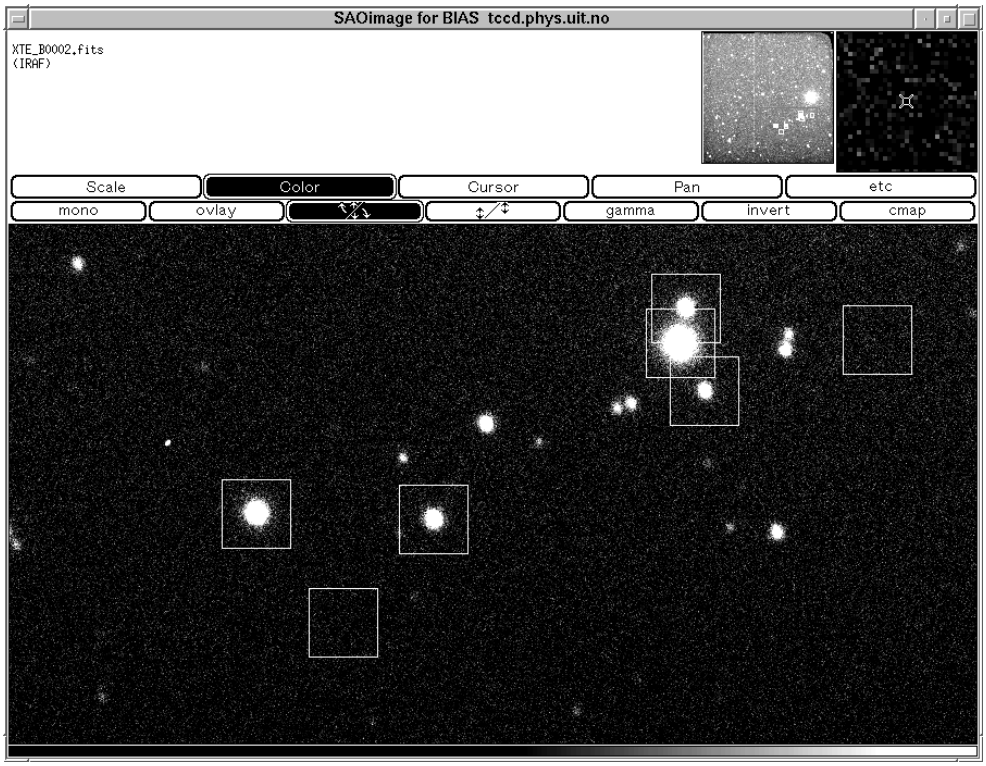


Figure 3.4: `saoimage` with some overlapping windows selected in the frame. The view here shows only a small fraction of a 2048^2 pixel image.

`expleft` counter is set to the exposure time, which should have been set with the `TIME` command. At last, the one millisecond timer interrupt is continued, and the exposure timer will start ticking toward zero.

`@FRAM` takes four digits from the input buffer, converts them to an integer value and sets the `frame_left` counter to this value. If the number is 9999 the camera will continue taking new frames until it is interrupted by an `@IBRK` command during integration. The rest of the command proceeds similar to the `@SINT` command, except that the `cam_mode` flag is set to indicate sequencing.

The `@CLRA` command is provided to directly erase the CCD through a call to the `imclr()` procedure, and the `@FRES` command will reset the format parameters of the CCD to their default values by calling the `format_reset()` procedure. The `@DOWN` command instantly kicks the camera out of whatever it is doing and jumps to the download procedure stored in the on-board PROM. Then the controller is ready to recieve and store new control software in RAM. This command is only used by the independent program `download`, described below.

Only two commands are used for initializing and activating windowed readout, the `@WINS` and the `@IMTX` command. The `@WINS` command is used to initialize the list of windows to read out, `wins`. The parameter takes a total of 18 characters from the

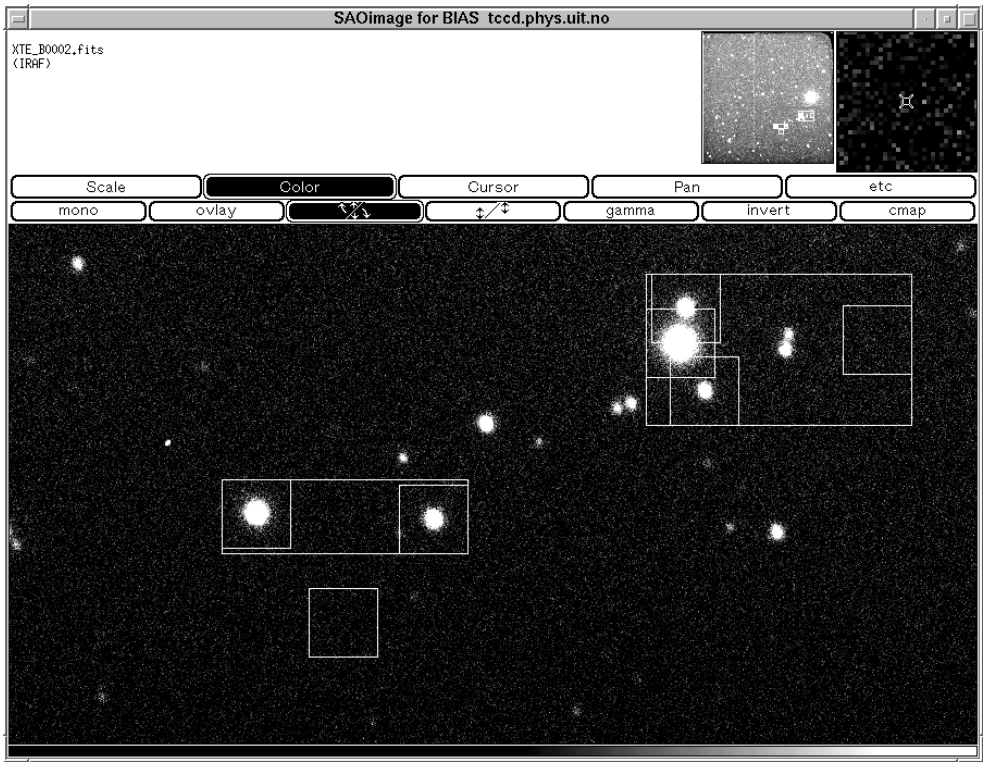


Figure 3.5: The same image as in Fig. 3.4, after `photinit` has calculated the actual image area to read out, and drawn the outer boxes around windows that overlap or lie along the same pixel row. `photinit` is described in the next chapter.

input buffer, to identify and initialize one CCD window. The first two characters identify the window number, and the subsequent 16 are divided into four groups of four which are converted to integer and stored as `xpos`, `ypos`, `xsize` and `ysize` for the specified window number. The number of windows is currently limited to 64, which should be more than sufficient. The `wins` array is 512 bytes long and is kept in memory outside the 1024 byte header info block. It is vital that the windows are properly sorted by the user computer system before windowed readout is started. There are two requirements to the sorting. Only one window can be present on each CCD line, and no new window can start before the previous is complete. Windows that overlap in the serial shift direction must be contained in one large window, as illustrated in Fig. 3.4 and Fig. 3.5.

The `@IMTX` command is used to switch between full frame readout and windowed readout. It's current value is reflected in the `im_tx` header field. A value of 1 indicates normal imaging and 2 indicates partial transfer, or windowing. A value of 0 implies that no image is transfered, although all pixels are sampled as normal. This is a leftover from the tip-tilt adaptive optics system of the HiRAC, where the image is processed in the camera controller memory to provide rapid correction factors for the adaptive system. It has no function in the current system for the TCP, but

is left as an option since it is quite simple to implement aperture photometry in the camera controller to produce an instrument that behaves similar to a multiple-channel photometer. For the moment we want to store all sampled data to use in our photometric analysis, and we leave this as an option for future implementation.

3.2.4 Clocking the CCD

The pixel readout algorithm is the core of the readout process, so we will start by introducing it before we show the particulars of the readout process. The purpose of this procedure is to sample a number of pixels that have been prepared on the serial output row of the CCD chip. The `pixel()` procedure is one of the very few parts of the original code that has not been attempted to port from assembly to C, to ensure that the precise timing of the original version is not compromised by the C compiler. A complete elaboration on all the details of this procedure will not be given, since it involves quite a number of steps. Basically, the procedure reads a count number preset before the procedure is called `count_x`, as well as the parameter `binx` that determines if any on chip pixel binning is required. (Binning is not used in windowed mode.) After initialising the needed registers, `pixel()` performs one complete sequence of serial shifts to put the charge in the serial register bin closest to the current output amplifier onto the summing well. If serial binning has been requested, further sequences of serial shifts are performed to place additional pixels onto the summing well. Then the last transition in the serial sequence is performed that isolates the charge on the summing well, and the charge conversion process can start. The first step is to start the clamp measurement to determine the base level of the output gate. While the clamp time `tcla` is ticking the conversion of the *previous* pixel measurement is transmitted to the user computer. Next, the summing well phase is turned off to let the charge proceed from the summing well to the output gate. The CDS sample phase is initiated, and after the sample time `TSAM` is out, the procedure starts from the beginning with reseting the output gate, and repeats until the last pixel has been sampled. Note that the last pixel measured has been measured, but not transmitted by the time the `pixel()` routine returns.

The total readout time for a CCD image can be written

$$T_{\text{readout}} = N_y \cdot (N_x \cdot (S_{\text{ser}} \cdot T_{\text{ser}} + T_{\text{cla}} + T_{\text{sam}}) + S_{\text{par}} \cdot T_{\text{par}}), \tag{3.1}$$

where N_x and N_y are the number of pixels in serial and parallel direction respectively, S_{ser} and S_{par} are the number of steps in each serial and parallel shift, T_{ser} and T_{par} are the serial and parallel settling times, and T_{cla} and T_{sam} are the clamp and sample times. In the header T_{ser} and T_{par} are equivalent to `tser` and `tpar`, and are given in CPU clock cycles of 50 nanoseconds. However, for the shortest times like `tser` which is given as 1 this is not accurate, since one has to add the number of clock cycles that executing the start/stop instructions, which gives the minimum times

the controller can handle, which is about 10 clock cycles. Equivalently, T_{cla} and T_{sam} are specified by `tcla` and `tsam`.

All these parameters are of course chip specific, and for the SiTe TK1024 chip we have the following values; $N_x = 1124$, $N_y = 1024$, $S_{\text{ser}} = 8$, $S_{\text{par}} = 7$, $T_{\text{ser}} = 0.5\mu$s, $T_{\text{par}} = 80\mu$s, $T_{\text{cla}} = 5.5\mu$s and $T_{\text{sam}} = 4.5\mu$s. Inserting these parameters into Eq. 3.1 gives us a total readout time for a full image of

$$T_{\text{readout}} = 1024 \cdot (1124 \cdot (8 \cdot 0.5 + 5.5 + 4.5) + 7 \cdot 80)\ \mu\text{s} \simeq 17\ \text{s} \tag{3.2}$$

In reality the time is slightly longer than this theoretical minimum, since the processing time of the CPU uses quite a few cycles while performing the operations. We have measured the readout time on our system to about 23 seconds.

It is necessary to erase the CCD before each exposure to clear any dark counts that has accumulated since the previous integration. This is performed by the procedure `imclr()` in `ccdsubs.c`. Between successive sequence exposures a full erasure would not be absolutely required, although it is of course essential to clear any unsampled pixels remaining on the chip after all the desired windows have been read (post sweeping).

Note that erasing all the data on the CCD in not an instantaneous operation, because even if no pixels are sampled the settling times for the shifts must be kept for all charge to be shifted out of the pixel wells. We can drop the intermediate serial shifts before each parallel shift, since we do not care about having a clean serial register before the lines are put there. Then we just need to do as many parallel shifts as there are lines on the CCD, and as many serial shift as there are pixels in each line to get all the accumulated charge to ground. The clearing time will then be (using TK1024 parameters):

$$T_{\text{clear}} = 1024 \cdot 7 \cdot 80\mu s + 1124 \cdot 7 \cdot 0.4\mu s \simeq 0.58s \tag{3.3}$$

Even if we do not perform any erasure between successive sequence frames, this number represents the lower limit of the readout time, *i.e.* a readout of zero windows. The number is a significant fraction of the readout times we hope to achieve with windowed readout, so we should do our best to reduce it. One approach would be to switch to dual mode readout during the clear operation. But parallel dual mode (AC-readout) only helps if we want to erase the whole CCD array. For erasing unwanted lines before the first window, between windows and after the last window changing the readout mode would only complicate matters. However, the `imclr()` procedure that must be called before the first integration starts, to clear out accumulated dark current counts, have been implemented using dual readout. The total number of parallel shifts are `toty/2 + PSHIFTX`, where the last parameter is a number of extra clears to get rid of all remaining charge. In `tk1024.h` `PSHIFTX` is set to 25, which is normally sufficient, but not enough to completely clear the CCD if the chip has

been saturated. The lowest clearing time we can reach with the TK1024 is therefore just under 0.3 seconds.

When reading out a real CCD image we also have to deal with overscan. The TK1024 has 50 overscan pixels on each side of the serial registers that do not correspond to pixels on the CCD image array. The CCD array is 1024×1024 pixels, but the serial register has 1124 pixels which we must take in consideration when reading out the image. In practice, the image size is set to a default of 1124×1024 pixels, and the controller treats this as if it was the real image size. Images produced by readout will then always have a 50 pixel overscan strip on each side of the image array, which is quite enough to determine a reliable bias level. For the Loral 2k chip, however, there are only two extra pixels along each edge, so that the effective image size is 2052×2052, although the real pixel array is 2048×2048. Two pixels are not sufficient to determine a reliable overscan level, so for Loral 2k chips it is recommended to use synthetic overscan. Synthetic overscan is produced by setting the serial clock drivers in *stuck* mode (by setting the `SER_ST` bit), *i.e.* the particular drivers do not respond to clock switching patterns, but keep a steady voltage. Then, a normal pixel readout can be performed, and will reproduce pixel values that accurately represent the zero level with the rms noise of the active output amplifier and signal chain.

3.2.5 Windowed readout

The windowed readout mode, introduced in Chapter 1 as our solution to reducing the readout times, produces readout times determined by the positions, sizes and number of windows selected by the observer. If all of the windows are square and of the same size, and none overlap or are located on the same line, we get a readout time of

$$\begin{aligned} T_{\text{readout}} &= N_y \cdot [N_{\text{sam}} \cdot (S_{\text{ser}} \cdot T_{\text{ser}} + T_{\text{cla}} + T_{\text{sam}}) \\ &\quad + (N_x - N_{\text{sam}}) \cdot S_{\text{ser}} \cdot T_{\text{ser}} + S_{\text{par}} \cdot T_{\text{par}}], \end{aligned} \tag{3.4}$$

where image size and timing parameters have the same notation and values as before, N_{sam} is the number of pixels to sample along a line or column (*i.e.* number of windows times size of window).

With the windowed readout method, the time lost to reading out the CCD can be effectively constrained by the observer when choosing the size of and number of windows. With more reference stars the total readout time will increase, and the observer must choose between a lower duty cycle or reducing the size of the windows. Large windows have the advantage of being more robust to tracking errors and seeing variations, as well as making it possible to determine a useful sky value around each star.

In Table 3.4 the measured readout times are given for some choices of channels and window sizes. Note that the window dimensions are not restricted to the choices

Table 3.4: Sample readout times

Channels	16×16	32×32	48×48	64×64
3	0.70 s	1.08 s	1.47 s	1.88 s
5	0.95 s	1.58 s	2.23 s	2.92 s
10	1.58 s	2.83 s	4.14 s	5.52 s

given in the table; any useful choice of dimensions will do, even rectangular ones. Note also that the time does not go towards zero when the window size decreases. There is a minimum time determined by the time it takes to erase the CCD, since all pixels must be shifted to ground before a new integration can start, which is about 0.24 s with the current chip. Also, in the times listed, a constant period of time has been spent at the end of the readout to determine the bias level; sampling 4096 overscan pixels adds another 0.09 s to the minimum readout time.

We can also see from Table 3.4 that doubling the size of each window along both axes will not quadruple the readout time as one could assume; the increase is only slightly more than a doubling. The reason for this is that once we start to sample a line, all pixels on the line has to be shifted out to prepare the serial register for the next line, while consecutive lines that contain no wanted pixels can be held on the serial shift register without a clearing cycle between. Since clearing one line takes about 7.5 ms, while reading it takes 22.3 ms, we find that reading n pixels takes about $7.5 + 0.013 \cdot n$ ms. Thus, as long as n is small compared to the full 1024 pixel line, increasing the number of pixels along the readout axis produces insignificant increases in the readout time. However, increasing the window size in the parallel direction requires that whole new lines must be cleared on the serial register, in effect almost doubling the readout time when doubling the height of the windows. This means that we can get quite a bit more pixels in each window, without significant increases in the readout time, if we use rectangular windows. However, the advantage is small since our apertures in general are circular, and has therefore not been implemented yet. In cases where we use sky annulus, and apertures which are much smaller than the windows, rectangular windows could give significant savings, by allowing more sky pixels close to the stars to be sampled without increasing the readout time.

3.2.6 Getting the timing right

When we want to do time resolved photometry it is essential that the sampling is even and precise, to avoid problems with the Fourier transform procedure. Since the CCD controller is a real time system, it is possible to start a sequence of exposures with a given exposure time and measure the effective sequence time interval afterwards. It will be the sum of the exposure time, readout time, a shutter delay,

and a few milliseconds of system overhead. In practical situations, especially in campaigns, it is desirable to have well defined sequence time intervals, say five, ten or twenty seconds. To accommodate this, the `seqtime` parameter set by the `SEQT` command has been introduced, as described in the section on camera commands. When the window readout procedure has completed the readout process it checks if the total time spent is shorter than the value given by `seqtime`. If this is the case it will wait the remaining number of milliseconds, before proceeding with the next frame in the sequence. Thus, to get proper timing for a sequence time of ten seconds, the exposure time must be set in such a way that the sum of the exposure time, readout time, and shutter delay is less than this value.

3.2.7 Timing errors

Since the CUO controller was not designed with timing hardware capable of keeping track of long sequences of observations, we expected some lag in our software implemented timing procedure. As mentioned, the controller is only equipped with a 12-bit millisecond counter that overflows every four seconds. Our implementation adds up the measured time after reading every window (or line when reading a full frame), but since there is a truncation error in every such operation we expect to lose an average of half a millisecond for each measurement, or about two milliseconds for each integration when running with four windows.

A software fix could keep the average error around one millisecond, but might in some cases increase the error. To eliminate the problem, proper timing hardware must be implemented in the next generation of controllers. This timing problem is not fatal, since the error for a given set of windows will reproduce quite accurately from one exposure to the next. The error for a given run can be easily measured afterwards by comparing the start times of the exposures at the beginning of the run with those at the end.

3.2.8 The readout process

When the integration time is out, the `main()` procedure calls the `readout()` procedure in `readout.c`. The shutter closing signal has already been sent by the timer interrupt procedure, so the first thing `readout()` does is to count the time set by `shutter_delay` to delay readout until the shutter is completely closed. Now, the timer register and the parameter `readout_time` is cleared to start tracking the readout time. For precision, an assembly routine `addtime()` is used to transfer spent time from the systems 12 bit timer register `TIMREG` to the 32 bit header field `readout_time`. The routine is called after every line when reading out a full frame, or after every window when reading out in windowed mode.

Before the actual readout starts, `readout()` sets the header field `frame_type` to either 2, 8 or 10 to signal a single frame, a full image sequence frame, or a windowed frame. This tells the program receiving the data that image pixels will follow immediately after the next header has been transmitted. The next step in `readout()` is just that. The transmitter is turned on and the 1k header block is transmitted by a call to `sendinfo()`.

Next, `readout()` decides if we are in full frame or windowing mode, and calls either `imout()` or `winout()` to handle the different readout modes.

For `imout()`, the readout procedure consists of determining the readout direction, and computing how many pixels must be cleared on each side of the region that should be read out based on the values of `beginx`, `beginy`, `imx`, and `imy`. Then the appropriate transmitters are activated and a quick range check is performed on the calculated shift parameters. Next, `parclr()` is called to dump any leading lines, and the procedure enters a loop over the number of lines to sample and transmit. This loop contains four steps; call `parshft()` to dump one or more lines onto the serial register, call `lineout()` to sample and transmit one line, call `addtime()` to keep track of the readout time, and finally do a check to see if an interrupt has been received from the user computer. After all lines have been sent, the remaining pixel on the sampler is transmitted, any lines not read out are cleared, and finally the transmitter is reset to primary channel transmission (in case multi-channel readout had been activated).

The `lineout()` procedure is trivial unless dual serial readout or synthetic overscan have been requested. If a partial frame has been requested in dual serial mode, and the requested region is not centered on the CCD, the procedure must shift the pixels on the serial register so that half of the pixels are on each side of the serial register split. Next, the unwanted pixels on the beginning of the line is dumped by setting `shift_x` to the value computed in `imout()` and calling `sershft()`. Then, finally, the `pixel()` routine is called to sample the appropriate pixels, now correctly positioned on the serial register. After the pixels are sampled and transmitted, `lineout()` determines if synthetic overscan is required. If so, the serial clock drivers are all set to `stuck` mode, so than no clock shifts are performed during readout. The `pixel()` routine is then called with the requested number of overscan pixels. When this completes, the the readout direction is reset to get back to normal pixel shifts. Before `lineout()` returns, it clears any remaining pixels on the serial register, to prevent the next parallel shift from loading into non-empty serial pixel bins.

The `winout()` routine is the only part of the CCD controller system which is not in any way based on the original assembly code. It starts by counting up the number of lines with non-zero window sizes found in the `wins` array, that has been initialised with the `@WINS` command, stopping at the first zero x-size found. Only single amplifier output has been implemented, so if dual mode readout is set the result is not predictable. The routine also does not consider on chip pixel binning, so the values of `binx` and `biny` have no effect during windowed readout. The main

loop in `winout()` is now over the windows defined by `wins`. The routine keeps track of the current vertical position in the frame, and uses the y-position of the current window to compute the lines to dump, `shift_y`, and calls `parclr()` to dump those lines. There is no way to get these lines back, once the charge has been shifted off the chip, so backward shifts are pointless. Therefore, the `wins` array must be sorted in the readout direction by the user computer. If an erroneous negative shift is discovered, no shift will be performed and the routine will continue readout from the current position. Next `wins()` enter an inner loop over the lines in the current window, loading one line at a time on the serial register. As in `lineout()`, the beginning of the line is cleared, the desired pixels are sampled by calling `pixel()`, and trailing pixels on the line are cleared to prepare the register for the next line. After all the lines in the window have been read, the readout time is updated by calling `addtime()` and a check for interrupt condition on the reciever is performed. After all windows have been processed in this fashion, the last pixel left on the sampler is transmitted, lines remaining after the last window is cleared, and a final call to `addtime()` determines the total readout time.

After `winout()` or `imout()` has completed the pixel sampling and transmission process, `readout()` disables the transmitter, stores the accumulated readout time before it is reset, and updates the frame counter `frame_total`. Next, if this was the last frame in a sequence, the system goes idle (`cam_mode = IDLE`) and the timer is reset and started to resume the normal interrupt and control loops. If more frames are pending (`frame_left > 0`), `readout()` computes the time spent on exposure, shutter delay and readout, and decides if the sequence time is out. If not, the procedure will sit in a delay loop until the time is out. Lastly, the shutter is opened and the exposure time counter is reset to start a new integration. Then the interrupt timer is reset and activated to resume normal integrating mode.

3.3 At the users end

The programs in the CCD and FASU controllers are initialised and take commands from the observer system computer. The software that runs on this end is written entirely in C++, using a bottom up approach. The current system still has the status as a development and debugging system, and not much effort has been put into making a user friendly graphical interface yet. Most of the effort has been put into making some useful C++ class libraries, that will be described in the following: These class libraries separates the different tasks into logical compounds: The `fits++` library takes care of reading and writing of FITS files and the handling of header keywords, as well as providing a data-type that makes arithmetic on arbitrary datasets as simple in C++ as in the high level language IDL. The `PCBoard` class gives an interface that implements basic read/write functionality to the fibre-optical I/O board, and the `Camera` class uses the `PCBoard` class to implement a simple interface to a CCD instrument connected to such a board.

The idea is that when all these have been assembled together in a working fashion it will be a reasonably quick job to create a beautiful user interface that calls these library functions without having to think about all the low level hardware specific matters.

For the moment, two programs are required to initialise and control the CCD camera system. The `download` program transmits the CCD sequencer control program described in the previous section to the sequencer computer, after which it will be executed by the CCD controller. Once this is done, and the FASU unit has been powered on, all system control is provided by a compact terminal driven program called `tcpcom`, which will be described at the end of this section.

3.3.1 Basic libraries

Opening the Buffer

A `PCBoard` class library has been implemented to handle all reading and writing through the optical interface board. It defines handles for the hardware registers listed in Table 2.17, and is initialised with the base address of the actual card.

The `init()` call will activate the reciever, clear the preset register, turn off autoincrementation and set the address pointer to the start of the buffer memory. The functions `rtxon()` and `rtxoff()` enables or disables the reciever and transmitter, `rxon()` enables the reciever and disables the transmitter, and `txon()` enables the transmitter and disable the reciever. The functions `incon()` and `incoff()` turns on or off the autoincrementation on the read and write pointers. `inconr()` and `inconw()` activates autoincrementation on respectively the read and write pointers only. `set_addr()` sets the pointer for read/write operations to an explicit address.

The procedure `clear()` will clear the requested number of words in the buffer memory, while the procedure `testram()` will attempt to count up the available buffer memory on the board and return the RAM size in megabytes.

Two version of the basic read and write functions are implemented. `read()` and `write()` activates autoincrementation and reads the whole block in one single operation, while `reads()` and `writes()` are slow versions that reads/writes one word at a time, with explicit incrementation of the address register between each operation. The last version avoids the problems observed on some PC systems. Note that write here means writing to the buffer memory, and is in no way related to actual transmission of data to the camera, since the buffer memory on the interface board is only used for recieving data from the camera end.

To send commands or data to the camera, the transmitter must be explicitly turned on and each word is transmitted with the `send()` command. After a sequence of data is sent, the transmitter is turned off. All the time while transmitting data the

reciever can be kept on (if `rtxon()` and `rxon()` is used to turn the transmitter on and off), thereby insuring that no data sent by the camera is lost while transmitting. The `wait()` call will wait for an acknowledge signal to be returned from the CCD controller to indicate that the character sent has been received and accepted, and should be used after each call to `send()`. `wait()` will eventually time out (after 5 million attempts), and return zero instead of one to signal an error condition. The actual time that it takes for a timeout condition to occur depends on the speed of the system.

Talking to the Camera

The details involved when using the `PCBoard` class to implement the communication with the CCD controller is contained in the `Camera` class library.

The basic commands are `getHeader()` and `getData()` to receive header information or image data from the camera, and `command()` to send a command line to the camera controller.

`getHeader()` reads the current 1024 byte header from the PC board buffer memory, and performs the necessary byte-order conversions. It also receives status information from the FASU, if available.

`getData()` will start a loop reading 256 byte blocks of image data into memory. Both fast and safe reading modes have been implemented. In any case, the procedure will search for a four word test pattern at the end of each block that is written into the buffer memory at the start of each integration. Only when the test pattern has been overwritten by the received image will the block be accepted, and the procedure will continue with the next block. To avoid overloading the computer with continuous reads to the buffer memory, a `usleep()` system call is executed every time a read is unsuccessful.

`command()` is the fundamental procedure for sending low level instructions to the CCD controller. It will turn on the transmitter, wait a short while for activation to be completed, and start sending characters the buffer string with appropriate `wait()` calls between. It will finish by sending a 'CR' character, after which it will turn off the transmitter. The `abort()` command is the simplest example of a `Camera` library function using the `command()` call, and is implemented with the single instruction: `command("@IBRK")`.

Other simple `Camera` commands are `expose()` and `dark()`. These are identical, except that `expose()` will activate the shutter if it is inactive, and `dark()` will do the opposite, ensuring that `expose()` will make an integration with the shutter open and `dark()` with the shutter closed. Both can be called with the integration time given as a floating point number, which are then converted to milliseconds before being sent to the camera with a `command("@TIME`*nnnnnnnn*`")` call. If no time or a negative time is given, the integration time will not be set, implying that the

same time as before should be used. After initialising the test pattern in the buffer memory, a `command("@SINT")` is sent to signal start of integration.

The command `printHeader()` does not strictly belong in the `Camera` class, as it is not concerned with communication to the camera, but rather with output to the user. It simply prints information pages based on the camera header to the standard output. The `makeFitsHeader()` call takes a Fits class variable as parameter, along with an optional filename, and generates proper FITS header keywords based on the current `Camera` header information. The filename parameter is inserted into the FITS header, along with information about the telescope, detector, date and time of observation, exposure timing information, information on the gain, readout direction, MPP mode, overscan and filter position. All header keywords inserted are listed in Table 3.5, together with the `Camera` parameter or call used to generate the value. In addition, the `fits++` class library automatically inserts the required information about image size and dimensions. However, not all the keywords have sensible values yet. For instance, information on the observatory and telescope is only inserted if this is explicitly compiled into the code (as has been done for the version for the NOT). The COMMENT and OBJECT field also contains no sensible information. It is also intended that the coordinates of the target along with sidereal time, airmass and other information provided by the telescope control computer should go into the header, but since the link to the telescope control computer has not been implemented yet, these keywords are tagged as 'Unknown'.

Storing the Data

Once the full image has been received from the CCD camera controller it must be written to disk in a useful format. The standard format used for astronomical images is the FITS format [Wel81]. It allows data of any type (byte, integer, float, double, etc.) and any dimensionality to be stored in a single file together with a header that can contain an arbitrary number of parameters or *keywords* describing the data set.

Since the software required for the windowed CCD photometry method uses images in various forms at all levels, extra effort has been put into generating an image data type for C++ based on the general FITS definition. `fits++` is a class library that provides a flexible and easy to use implementation of a `Fits` data type.

The `Fits` objects are constructed in several layers. The most fundamental class is the `Fitsio` type. An object of this class contains the same information as is always contained in all FITS files; the header and data with information about data type, number of axes and number of points on each axis. The `Fitsio` class contain all necessary functions to read and write FITS files, including functions to convert between different data types and functions to construct and edit the FITS header. Arithmetics on `Fits` objects in C++ require that the object is of a well defined data

Table 3.5: FITS header keywords added by `makeFitsHeader()`.

Keyword	Value	Comment
ORIGIN	TCP-Test	
OBSERVAT	No default	
TELESCOP	No default	
INSTRUME	TCP V1.0	
DETNAME	CCD8	ccd_type
DATE	`st_date()`	
DATE-OBS	`st_datetime()`	System UT at start
FILENAME	`fname`	TCP image file name
OBJECT	-	-
EXPTIME	`exptime`	Exposure time [s]
RO_TIME	`readout_time`	CCD read-out time [s]
SEQTIME	`seqtime`	Time between sequence frames [s]
EXPNUM	`frame_total`	Sequence number
TM_START	`st_sec()`	(HH:MM:SS) UT start time [s]
TM_END	`en_sec()`	(HH:MM:SS) UT end time [s]
COMMENT		Comment line
GAINM	`cdsgain`	High or Low
AMPLM	`readdir`	Active amplifier(s)
CCDTEMP	`ccd_temp`	Chip temperature at start [C]
LN2TEMP	`ln2_temp`	Cold point temp at start [C]
PRESSURE	`pressure`	Camera head pressure at start [mb]
MPP	`mpp_mode`	Multi pinned phase option
CHIPID	`ccd_type`	
DATAMIN	`im.min()`	Minimum data value
DATAMAX	`im.max()`	Maximum data value
XOVERSC	`over_scan`	Number of overscan pix after line
YOVERSC	0	Number of overscan lines after image
AOVERSC	0	Number of overscan pixels before line
FILTER	`filter`	Filter wheel position
ST	Unknown	Sidereal time at start
RA	Unknown	Right ascension at start
DEC	Unknown	Declination at start
AIRMAS	Unknown	Airmass at start
ROT	Unknown	Rotator position at start
FOC	Unknown	Telescope focus at start

Table 3.6: FITS types implemented in `fits++`.

BITPIX	C type	Fits type	Status
8	char	N/A	Convertible
16	signed short	N/A	Convertible
16	unsigned short	`Fitsu`	Good
32	long	N/A	Convertible
64	long long	N/A	Convertible
-16	N/A	N/A	Not convertible
-32	float	`Fitsf`	Good
-64	double	N/A	Convertible

Table 3.7: Fits++ header manipulation routines.

Command	Param.	Opt. par.	Description
GetKeyword()	kwd, val	comment	Retrieve value and comment
InsertKeyword()	kwd, val	comment	Set keyword and comment
DeleteKeyword()	kwd		Remove keyword from header
InsertHistory()	value		Insert a new history field
DeleteHistory()		comment	Delete history field starting w/comment
UpdateHeader()			Update format kwds to current format

type. Therefore, a number of subclasses is derived from the `Fitsio` class to provide the desired arithmetics for each class. To declare a `Fits` object with data points of type float, a template type declaration is used. For example, to declare an object called image of type float, use: `Fits<float> image;`

A short code example;

```
Fits<float> image;           // Declare fits object image
image.read("data.fits");     // Read the file named data.fits
image = image - min(image); // Offset zero point to minimum value
image.write("test.fits");    // Save new image as test.fits
```

Note that the original file `data.fits` can be of any type. The data will be converted to floating point precision when its read, and all arithmetics will be performed with this precision, regardless of the original type. The saved file will be of the declared type, *i.e.* float.

The FITS standard allows for integer data types with 8, 16, 32 and 64 bits per pixel and floating point types with precision 16, 32 and 64. The `Fitsio` class can handle most of these types, but not all of them have been implemented as subclasses

Table 3.8: `fits++` basic commands.

Command	Parameters	Return value
`bitpix()`	-	FITS BITPIX equivalent
`dsize()`	-	Number of bytes of data type [1,2,4,8]
`naxis(), dim()`	-	Number of axes
`npix()`	-	Total number of pixels in dataset
`nbytes()`	-	Number of bytes = npix()*dsize()
`naxis1(), width()`	-	Number of pixels along 1st axis
`naxis2(), height()`	-	Number of pixels along 2nd axis
`naxis3(), depth()`	-	Number of pixels along 3rd axis
`size()`	-	IDL-like array of dimensions
`read()`	filename	Read FITS file
`write()`	filename	Write FITS file
`clear()`	-	Clear all pixels in image

with well defined arithmetics, as indicated in Table 3.6. 8 bits data could easily be handled with the char class, but doing arithmetics on chars is an endless row of over- and underflows, so it has simply not been implemented. 16 bits floating point has no counterpart in C/C++ and is therefore not supported. The remaining data types correspond to `<short>`, `<long>`, `<long long>`, `<float>` and `<double>`. `<int>` and `<long int>` are equivalent to `<long>`, although `<long>`, `<long long>` and `<double>` remains to be implemented. Note that the `Fitsu` type corresponds to `<unsigned short>`, which is written to FITS files setting the BZERO keyword to indicate a zero point offset of 32768, as is the conventional way of storing 16 bit CCD images.

The `Fitsio` class, and all derived classes, contain a FITS header. Header pages are actually objects of a separate fundamental class (`Header`), and are organised as a linked list of header pages, each containing 36 lines of 80 characters (2880 bytes), as the FITS standard specifies [Wel81]. Dynamic functions allow retrieval, replacement and deletion of all keywords, and new header pages are added and removed in a way that is completely transparent to the user.

A number of functions on the basic `Fitsio` class returns information about the properties of the dataset. `bitpix()` returns the number of bits per pixel of the data, positive for integer type and negative for floating point. `naxis()` or `ndim()` returns the number of axes or dimensions of the data set, `npix()` return the number of pixels and so on.

The fundamental input/output functions `read()` and `write()` both take the file name as argument and performs the required operations to read or write the FITS header and dataset as defined.

For most computations on data based on FITS files, floating point precision is desired. Even if the original data is of integer type, most calculations can conveniently be

done as floating point, and then converting back to integer before output, thereby avoiding unnecessary over- and underflows. `Fits<float>`, or `Fitsf` for short, is also the most developed subclass of the `Fits` type.

All arithmetics on `Fits` class objects are performed with the standard operators, just as if the data were normal variables. Addition, subtraction, multiplication and division will work as one should expect both for binary operators (`+`, `-`, `*` and `/`) as well as unary operators (`+=`, `-=`, `*=` and `/=`).

It is very important for programmers who want to use the `fits++` class to understand how statements are evaluated by the compiler. There are many ways to write a statement that will produce the same result, but with very different performance. Just like in ordinary C/C++ a lot can be gained by using unary operators as much as possible, since they do not require any temporary storage. Consider the two statements:

```
im1 = im1 + 25;
im1 += 25;
```

The first statement contains two operators, the `+` operation being evaluated first and then an assignment. The `+` operation will loop through the dataset and add the number 25 to each pixel, but the compiler is normally not smart enough to realize by itself that it no longer needs the original data after the addition, so it will create a temporary storage as large as the whole dataset to hold the sum. The assignment will then loop through this temporary dataset and copy it back over the original data. The unary operator loops just once through the data, performing the addition on each pixel without any need of temporary storage.

Many sub-classes can be defined based on the different circumstances FITS images are used in. For our purpose we have defined a sub-class called `CCDimg`, that takes care of a few basic things relevant for CCD images. It defines overscan sections, procedures for removing and keeping track of bias levels, integration times, start and stop times and such. Another class called `CCDarr` defines an array of objects of the `CCDimg` type, together with the procedures necessary to keep track of dimensions and conversion. These classes are essencial in the processing of the window frame images.

3.3.2 `download`: CCD system start

When the CCD controller unit has been powered on, it starts running the program that is contained in its EP-ROM memory. For our development system this has been an EP-ROM module that contains a download procedure initialised by sending a `@DOWN` command to the controller. The small utility called `download` takes care of this, by sending the `@DOWN` command followed by the code for our windowed readout

software, in S-record format. When we compile the software with the `gcc` cross-compiler, using the `make srec` command, a file named `im.srec` is produced. When download is called, the path to this file must be given as a parameter. The final version of the controller code should be burned to a new EP-ROM and installed on the controller board, making this program only useful in future development, and not for normal operation.

3.3.3 `tcpcom`: The basic control program

`tcpcom` is a simple terminal window based interface to the Copenhagen CCD controller system, with particular extensions developed for the TCP windowed CCD readout mode. It is not intended as a general user interface to the instrument, but more as an effective debugging tool that gives you access to all the informations and resources that the camera controller system provides. The display shows the internal variables in the camera controller, using the source code names as identifiers (of which not all have meaningful names). The `tcpcom` code declares one object of the `Camera` class, called `Cam`, that will refer to the CCD camera as connected through the fibre I/O board.

`tcpcom` uses the `Camera` class definitions to read the header information with the `Cam.getHeader()` call in a 100 ms loop.

Reading exposure frames

When the camera expousure time is out, the camera signals a new frame that automatically resets the buffer cards write pointer to the start of the buffer memory. A standard header information page is then sent with the `frame_type` flag set to `READOUT`. Immediately the pixel data values will follow, as each requested pixel are sampled. Now, the transmitted data are all stored in the buffer memory on the interface board, but `tcpcom` must read and store these images as they arrive, or the next incoming image will overwrite the previous. For `tcpcom` to know that we do not read ahead of the new data arriving in the buffer, it is necessary to have prepared the buffer memory before readout commences. The `Cam.prep()` call writes a distinct two word pattern at the end of every 256 byte block. The code in `Cam.getFrame()` then reads the frame block by block, waiting with accepting each block until the control pattern has been overwritten.

The design of the buffer card is such that all data is stored and kept on this board until it is overwritten by the next frame. Thus, if an error occurs during the readout of an exposure it is quite possible to recover the data by just restarting the readout.

A problem will occur if the camera commences readout of the next frame before the previous image was safely stored, because the last thing the software must do is to initialize the the PC-board buffer. The `getImage()` procedure in `tcpcom` checks

that a new image readout has not already started before it calls `Cam.prep()` to avoid overwriting data already sent by the CCD controller, but if this is the case there is no warranty that the incoming image is correctly read. Without the test bytes written into the buffer memory by `Cam.prep()`, `Cam.getData()` will not be able to determine what is new data and what is from the previous image, and will return inconsistent data. No such behavior has been detected, though, since the user computer system is much faster than the sequencer computer. Under very heavy load, such problems can occur, so it is strongly discouraged to use the user computer for any heavy data processing during readout.

Programming the Camera

The `tcpcom` program can also be used as a simple command interface to the camera. By hitting the '@' key the header display loop halts, and a command line starting with '@' is given at the bottom of the screen. Now all basic commands as given in Table 3.3 can be sent directly to the camera. If the line is successfully interpreted by the camera software, the effect should show up immediately in the next header page received. This is a rather awkward and non-intuitive way of using the instrument, but remember that `tcpcom` was only meant for testing and debugging of the instrument. All frequently used commands have been implemented as keystroke shortcuts in `tcpcom`. For instance, the 'W' key reads the list of windows in the file `ccdwins.dat` and transmits it to the camera as a sequence of `@WINS` commands. The list of keystroke commands are seen in the `tcpcom` on-screen help page, shown in Fig. 3.6.

Setting up `tcpcom`

The PC-board talks over port addresses in the range 0x500 – 0x55F. To access this range under Linux it is necessary that the program is run with super user privileges. That is; either it must be run by root, or the program must be owned by root, and the *suid bit* (set user id) set on the executable file.

If communications through named pipes with an external filter wheel driver is required, the environment variable `DEV_DIR` must be set to point to the directory that contain the named pipes, as is the case when the HiRAC filter wheel is used.

If the CCD controller uses a download EP-ROM for the camera controller software, the download program must be run before `tcpcom` is started.

Running `tcpcom`

The program normally does not require any options or environment variables to run. An exception is if the fibre-optical PC-Board interface is located on a different port

number than the default 0x500. Then the option string `--port <number>` should be given, where `<number>` is one of 500, 520 and 540. If `tcpcom` is run with a controller code that does not support checksums (like the standard CUO controller) the option `--debug` must be given, to prevent the program from terminating with an error message.

The `tcpcom` screens

The `tcpcom` terminal screen is divided in two, where the upper half gives the most basic information about the camera status, and the lower half gives several pages of more or less useful information. The information on the lower half is changed by using the number keys; the '0' key gives some standard info, the '1' key gives timing related information, the '2' key gives description of voltages specifics, the '3' key gives some internal `tcpcom` information (save file names, counters and PC-Board read modes), the '4' key may give information about filter wheel and telescope status if this is available, and finally the '9' or '?' key gives a help page as shown in figure 3.6.

Basic operation

All commands to the camera can be sent as '@' commands, as described in the previous chapter. For instance, the command `@TIME`*nnnnnnnn* will set the exposure time in milliseconds, and `@SINT` will start an integration with this time. All the most used commands have been implemented as keystroke shortcuts. For instance, the 'E' key will ask for an integration time (in seconds), send the appropriate `@TIME` command to the camera, and start an integration with `@SINT`. The 'e' command immediately starts an integration with the previously set integration time.

To make bias and dark frames the automatic shutter must be turned off. This is done with the `@TASH`' command. Normal `@TIME` and `@SINT` commands will now produce bias and dark frames. The keystroke command 'b' sets the exposure time to zero, toggles the shutter and starts an exposure, and the 'd' command asks for a exposure time for the dark frame, and initialises this. Note that after completion of any of these commands the automatic shutter is switched off, so that any integration started with the low-level `@SINT` command will do a dark frame. The 'e' shortcut assumes that you want a non-dark exposure and will send a `@TASH` command if the auto-shutter option is turned off, as can be easily confirmed as the variable `auto_shutter` on the display will change when these commands are given.

When using low level commands to start integrations with the camera, it is important to realize that the PC-Board memory needs to be initialised if `tcpcom` shall be able to track the arrival of new data from the camera. The memory is prepared for a new image with the 'p' command, and this command does not give any feedback.

Figure 3.6: `tcpcom` display with help page.

```
-------------------------------------------------------------------------------
ID: BROR  CCD: TK1024    VERSION: Roy v1.0        UT: Sun Sep 23 22:30:16 2000

MODE: IDLE               DATA: HEADER             TRANSFER MODE: FULL FRAME
totx             1124    toty             1024    ccd_temp        24.11 C
beginx              1    beginy              1    ln2_temp        24.62 C
imx              1124    imy              1024    ref_temp      -100.00 C
binx                1    biny                1    pressure          N/A
exptime          1.00 s  expleft          0.00 s  seqtime          0.00 s
frame_left          0    frame_total         0    readdir             C

TCPCOM Keystroke commands:
 e: start Exposure                  E: set exptime and start Exposure
 d: start dark Exposure             b: start Bias exposure
 s: start exposure Sequence         S: set Sequence time interval
 a: Abort exposure or Sequence      D: set data Directory name
 w: read Windows definition file    W: set Window filename base
 f: set Filter                      F: set Filter sequence
 A: re-init filterwheel and shutter H: toggle safe/fast Header read mode
 l: start temperature Logging mode  R: toggle safe/fast data Read mode
 t: toggle Transfer mode            1: display Timing specifics
 N: set Filename base               2: display Voltages setup
 ?: display this help page          3: display tcpcom internal System variables
 #: toggle Debug option             4: display filter and shutter info
 @: send raw commands to camera     0: set Normal display mode
-------------------------------------------------------------------------------
```

When using the keystroke commands, the memory is always initialised automaticly. Normally, the memory will be correctly prepared for a new frame as soon as one image has been completely retrieved.

For basic camera operation, the commands 'e', 'E', 'b' and 'd' are all that is required to perform observations with the camera.

The 'a' (abort) command sends an `@IBRK` command to the camera, which normally causes the camera to break an integration and go idle.

For our prototype instrument, the filter wheel is fully controlled by `tcpcom`, through calls to the FASU class library. For the systems at NOT, HiRAC and ALFOSC, special driver programs run on the user computer that is interfaced with `tcpcom` through unix pipes. `tcpcom` can position the filter wheel (and grisms for ALFOSC) and get information back on the current positions. The 'f' key will prompt for a filter number and set the filter wheel to that position. The current position should be reflected on the interface display, and in the FITS headers.

Sequence frame images and windowing

Sequence frame photometry is controlled by the 's' and 'S' commands, which corresponds to the low level `@FRAM`*nnnn* and `@SEQT`*nnnnnnnn* commands. The `@FRAM` command will start up a sequence of *nnnn* integrations, and setting the number nnnn to 9999 will cause the camera to produce images indefinitely (until an abort command is sent). The 'S' command will ask for a sequence time (in seconds) which will be converted to milliseconds and tranmitted to the camera with a `@SEQT` command. If a sequence time interval which is larger than the sum of the exposure time, readout time and shutter delay, is given, the camera will wait until the requested sequence time is reached before starting the next integration.

To do windowed photometry it is necessary to initialise an array of window positions in the camera controller by sending a sequence of `@WINS` commands. This is done automaticly with the 'w' command, which will read the `photwccd.dat` file produced by the `photinit` program. Once the windows have been initialised, the camera is switched to `PARTIAL FRAME` transfer mode, as opposed to the default `FULL FRAME` transfer mode, and all exposures and sequences of exposures will produce windowed frames. To toggle between full frame and partial frame readout, use the 't' key. We will discuss the details on how to create these window definition files in the next chapter.

Save file names

The full image frames and windowed frames have different file name bases. Refer to the system specifics information page ('3') to see the defaults. The file name bases can be changed with the 'N' and 'W' commands for normal and windowed frames respectively. The 'D' key will ask for a new data directory name, create that directory if it does not already exist, and all further data files will end up in this subdirectory.

3.4 Want to Observe

Now that we not only have assembled the hardware for our CCD photometry system, but have software running both in the CCD and FASU controllers, as well as a program to set up and recieve image and windowed photometry data, we can actually be on our way to the telescope and start some observations. However, we will not get much useful information from the data we obtain. At least not if the observations are made in windowed mode. In the next chapter we will describe the software for converting and processing our CCD data to produce useful astronomical information.

Chapter 4

The Windowed View

In the previous chapter we have seen how the camera controller program has been modified to handle windowed readout, and we have explained how `tcpcom` sets up the windowing mode by transmitting a set of coordinates that define the regions on the CCD chip to read out. But whereas `tcpcom` provides the control interface to the CCD camera and filter wheel, and facilitates automatic storage of the CCD data produced by the camera, the task of producing meaningful photometric measurements from these data files is left to the *Real Time Photometry* program, `rtp`, which is described in this chapter. A number of helpful applications that have been implemented around the two main programs `tcpcom` and `rtp` are also described here.

One extremely useful tool is the `photinit` program. It makes the generation of the initialisation files `tcpcom` uses when setting up windowed readout mode a very simple task, by interacting with the image display program, `saoimage`, to show an image of the field observed and allowing the observer to identify the targets and background fields by simply clicking on the image.

Fig. 4.1 shows how all these programs interact with each other and the image display program `saoimage` and plotting program `gnuplot`. In this chapter we will present these programs in a way that will be more for the astronomical users benefit than from a developmental point of view. Unlike the code in the two previous chapters, the code for these user end programs does not perform any hardware specific interactions, and is therefore quite trivial to interpret by reading the code. Thanks to the simplified image handling provided by the `fits++` class library, it is also quite compact and transparent.

4.1 `photinit` and the window definition lists

To do windowed photometry it is necessary to know the precise positions of the target stars on the CCD detector. This is accomplished by first obtaining an image

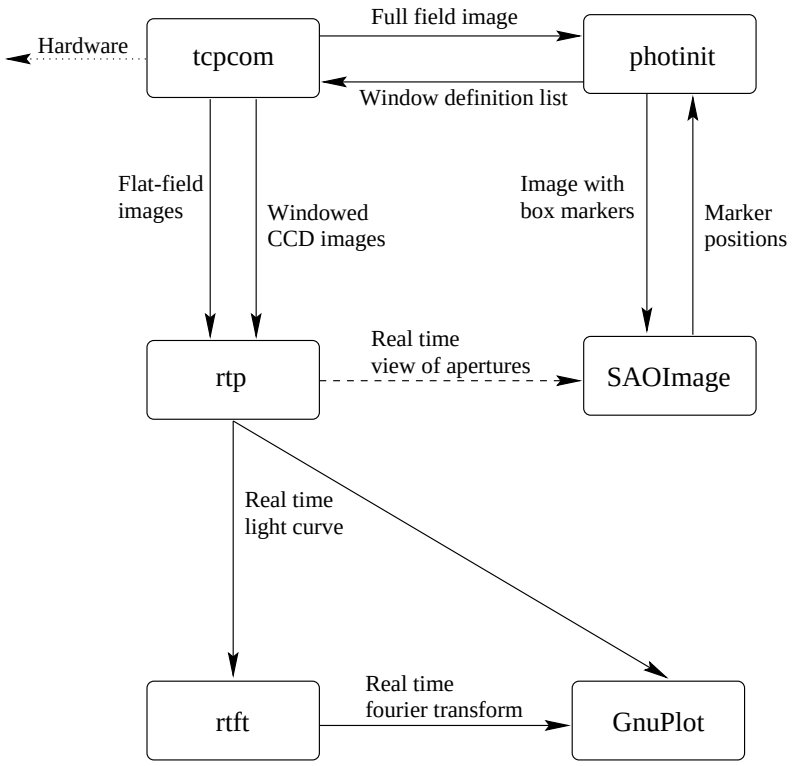


Figure 4.1: Real-time photometry software interactions.

of the whole field (with `tcpcom` in normal full frame imaging mode), and making sure that the target and reference stars are present in the field. Starting `photinit` will display this image (using the image display program `saoimage`) and allow the user to mark the relevant targets and sky fields. When this is done two files are created. The first, `photwins.dat`, contains the coordinates and dimensions of the fields in the order selected by the user, assuming that the primary target is the first (channel #1), followed by one or more reference stars, and one or more sky fields at the end. The second file, `photwccd.dat`, contains the images sorted in the CCD readout direction, and with a bias level field added at the end. It is this second file that contains the information that `tcpcom` reads and transmits to the camera to initialise windowed readout. The other one is used to reconstruct the correct sequence during processing with `rtp`.

`photinit` is called with the name of a FITS file containing the image of the precise field that the telescope is looking at and in which the target star has been located. For `photinit` to work together with `saoimage` it is necessary to set up the named pipes for communication and give the path to these on the command line when both programs are started. Since this is rather awkward, a script called `phot` has been made that takes care of this. The named pipes must exist at the locations given in the script for this to work, of course.

4.1.1 Setting up phot

photinit requires two-way communication with saoimage to allow the user to draw windows on the image display for marking off the photometry fields. For this to work the appropriate *Unix* pipes must have been set up in the directory referred to as DEV_DIR in the phot script. DEV_DIR can be /dev or $home/dev depending on whether a system installation or a local installation is preferred. Three pipes are required; one for input to gnuplot, and two for input and output to saoimage. If DEV_DIR is appropriately set, they can be created with;

```
mknod $DEV_DIR/gplot_i p
mknod $DEV_DIR/imt1i p
mknod $DEV_DIR/imt1o p
```

When the required named pipes have been correctly set up, photinit is ready to run. The normal way to start the program is through the phot script. This script will set up the required environment variables to start saoimage and photinit in such a way that they will communicate through the named pipes. The script will also kill any other process running saoimage to avoid display and pipe conflicts. Be aware that saoimage tends to crash if it is unable to allocate the necessary colors, for instance if Netscape or other color hungry applications are running. A new version of saoimage exists that is slightly more tolerant to display configuration problems, but this can not replace the version of saoimage that is used with photinit. The version used here contains particular extensions for use with 16-bit CCD data that the official saoimage does not have. It is a further extension of the version modified by J. Klougart of the CUO for use with the standard CCD image aquisition program for Copenhagen cameras, bias.

4.1.2 Starting phot

phot is invoked with a single parameter; the name of a CCD image to be used as a background for drawing the photometry aperture windows on. The data in the file are not actually used for anything, except as a guide for the user to orient the location of the windows, so for testing any FITS image of the correct size will do.

4.1.3 Commands in photinit

When the phot script has sucessfully started saoimage and photinit, keystroke and colon commands can be given when the pointer is placed in the saoimage display window. Keystroke commands are single character commands immediately processed, while colon commands enables a command line in the terminal window where phot was called, in which commands can be typed.

```
------------------------------------------------------------------

                        Help Page

Colon Commands:
    :file           - load new fits file
    :size <int>     - set square box size
    :status         - print current settings
    :print [file]   - print frame buffer to printer or file
    :prm            - print markers

Keystroke Commands:
    ?  - Print Help             q  - Quit
    b  - Box (rectangular)      s  - Square box
    d  - Delete marker          r  - Redraw markers
    D  - Delete all markers     L  - Load image
    M  - Load markers file      R  - Redraw image
    W  - Write markers file

------------------------------------------------------------------
```

Figure 4.2: Help page for the `photinit` program.

The '?' key produces a help page on the terminal as shown in Fig. 4.2.

The basic commands are 's', to place a square window around the area pointed to by the cursor, and 'W' to write the windows to output files that can be sent to the camera and used for processing the images. The order in which the fields are selected is important. `rtp` assumes that the first window contains the object star – channel #1 – followed by some reference stars, and that all sky fields comes at the end. The windows will be stored in this order order to the file `photwins.dat`. These windows may overlap in any way that is useful to the observer. The window list is then sorted and converted to a list of windows with no overlap in the readout direction, as required by the camera controller, and stored in the file `photwccd.dat`. The information in these files is required for any program that want to reconstruct the original window sequence from the output pixel stream the camera produces.

The 'M' command will read the currently saved window description files and draw them on the display. The 'd' key will attempt to delete any windows which the pointer is located inside of, although it may not be completely sucessfull in redrawing the display in all cases, but then the 'R' key can be used to redraw the image and markers. The 'D' key allows the user to start from scratch without any markers, and the 'L' key will reload the input FITS file if things get completely out of hand. The 'b' command draws rectangular boxes of arbitrary size, but such windows are

not useful for the processing in the current version of `rtp`.

The colon commands lets you input strings from the terminal window, to read a new input FITS file, change the default size of the square aperture windows, and a few others. Expecting frame buffer printing to work may be a bit optimistic, though.

4.2 Real time photometry

As mentioned, the real time processing of the data is performed by the `rtp` program. After the window definition files have been successfully generated and `photinit` terminated, `rtp` can be started. It will then read the window definition files and process the window frame FITS files as they are stored by `tcpcom`. But before `rtp` can be expected to work it needs to know where to find the relevant data files.

4.2.1 Setting up `rtp`

To process data with `rtp` and the accompanying utilities it is essencial to set up a working directory first. This is done by creating a direcory at any desired location and creating a symbolic link in this directory to the actual directory containing the data.

If you are doing real time photometry, simply create a directory where you want to keep the processed data, so that you do not overwrite any existing files, and make a link to the data directory where `tcpcom` stores the incoming CCD frames. For instance

```
> mkdir run1
> cd run1
> ln -s ~tcp/data data
```

As mentioned, `rtp` requires the files `photwins.dat` and `photwccd.dat` to run. These files should normally be in the data directory, but they can also be in the current directory. The last possibility will take precedence, if both exist.

If you want flat field processing of the data, make another symbolic link named `flats` to the data directory containing the correct flat field images. It may be the same as the previous data directory, for instance

```
> ln -s ~tcp/data flats
```

With this approach you do not need any write permissions to the data, and no additional files will be put in the original data directory during processing. It also works fine for post-run processing when the data are stored on a CD-ROM.

4.2.2 Running rtp

When rtp starts it will first sum all FITS files in the flats directory that have file names matching the default string flat*.fits. If you want to specify another query string for the flat fields, start rtp with the -f <flatstr> option. The flat fields that match the query will be averaged and saved as Flat.fits in your working directory. Next time you run rtp, unless you give a specific filename or query string, this file will be used.

rtp can be started and stopped at any point during or after the observation run. When it is started it will get the flat field and prepare the output data files, phot.raw, phot.dat and phot.dif, before searching for input CCD FITS files in the data directory. It will immediately process all the files it finds, and then go into a delayed loop where it will process files as they arrive.

rtp can also be used for more advanced processing of the data after the original session is over. Then it can sometimes be useful to have a delay between the processing of the images, to monitor the progress. This can be done by calling rtp with the -d <delay>, where the delay time is given in seconds and can be a floating-point number. A number of more advanced options such as automatic aperture centering and sky annulus processing can also be invoked when rtp is started. It is reccomended, however, to keep things simple for the real time mode in order not to put too much load on the computer while it is recieving and storing frames from the CCD camera.

4.2.3 Preprocessing

The flat field image, generated as described above, will be converted to a window list format by extracting the proper areas using the information in the file photwins.dat. During processing, each observation frame will be divided by the appropriate flat field window.

A bias level value is computed from the last window in the raw CCD image if there is information about such a window in the file photwccd.dat. This window is identified by having a height of one. Each CCD window is flat fielded and bias level subtracted before the aperture sums are computed.

4.2.4 Aperture photometry

rtp will process the CCD images using simple aperture photometry to sum up all the counts inside a predefined aperture. It is important to remember that the first few columns read from the CCD are not useful. In particular, the first pixel read in a line is always zero, the next pixel has a somewhat high level, and then the sampler gives progressively more reliable output as more pixels are sequenced. The default setup avoids the first four coloumns, using an aperture mask with a

radius of 30 pixels, centered at pixel [34, 30], when the image size is at the default 64x64. The aperture can be set from the command line with the `-r <radius> -x <xcenter> -y <ycenter>` options. Alternatively, it can be set interactively while the program is running. Change the radius and positions by using the `+-[]{}` keys on the keyboard, then hit 'M' to produce the new mask. Hit 'R' to reprocess all data from the beginning with this new mask.

Another way is to use the built in automatic centering algorithm. This computes a simple geometric center of gravity within the aperture, and recenters the aperture iteratively, a procedure that in most cases works very well. Automatic centering is described in detail below. For real time photometry, a fixed, large aperture is recommended. When processing the data offline later, different aperture sizes and centering methods can be attempted to limit the noise.

The aperture masks are generated with the same dimensions as the aperture windows, and are computed so that masking is achieved with a simple pixel for pixel multiplication with the aperture window. Thus, pixels inside the aperture have the value 1, and pixels outside 0. Significantly, the masks have soft edges, meaning that pixels that fall on the edge of the aperture will contain the the fraction of the pixel area that falls inside the aperture. This ensures that the mask can be centered with very fine resolution, and moved with sub-pixel steps, without round-off errors. If desired, the constructed aperture masks are saved in the working directory as `mask`N`.fits`, with one mask for each aperture, N. This requires running `rtp` with the debug option turned on. A test image of the current frame as seen through the aperture mask can be created with the 'W' key. The filename for this image will always be `testview.fits`.

4.2.5 In the Centre

The geometrical centre of an image used in the centering algorithm is simply the geometrical moment of the distribution defined by [Teu93, p. 157]:

$$b_x = \int\int x b(x,y) \mathrm{d}x\mathrm{d}y \tag{4.1}$$

and

$$b_y = \int\int y b(x,y) \mathrm{d}x\mathrm{d}y, \tag{4.2}$$

where $b(x,y)$ are the normalised pixel values. These integrals (or sums) give a useful centre (b_x, b_y) of the pixel intensities, but are taken only inside the previous aperture position, to avoid any neighboring stars that might be present around the edges of the window.[1] This means that if the star is off centre when the geometrical moment is computed inside the aperture, the new centre will go in the right direction, but

[1] The case of PG 1618+563 A & B in Ch. 6 is an example of a case where this is vital.

not all the way to the centre. A few iterations of computing this moment inside progressively better centred apertures will quickly converge on a very accurate centre position.

Note that this algorithm does not make any assumptions about the shape of the stellar profile, unlike most centering algorithms by astronomical CCD photometry software. Other common algorithms either fits a two dimensional polynomial through the star and computes the zero point of the derivative of this polynomial, or they attempt to fit some profile (*e.g.* a Gaussian) to the stellar image. We have avoided this here, since it is sometimes necessary to defocus the telescope somewhat to avoid too much of the light being confined to a few pixels, in which case the peak may no longer be the centre, and Gaussian profiles can no longer be assumed.

The desired centering algorithm is specified on the command line when `rtp` is started. The `-O` will compute initial offsets for each star, and the `-M` option will invoke the *Moving Aperture Photometry* (MAP) option, computing a new aperture centre for each frame, and recompute the aperture mask used for computing the aperture sums. For both options the `-i <num>` options can be given to specify the number of iterations desired.

Both the iterative recentering and the recomputation of the aperture masks with the soft edges takes a significant amount of computing time, compared to the plain aperture sums required for simple processing. It is therefore not recommended to use these options when doing real time photometry, and rather reprocess the data later trying out what options are required to produce the best possible results.

4.2.6 Making the Difference

The differential photometry for the target d_f for each observation i is given by subtracting the counts of the reference star n_r, scaled to the mean level of the target $\overline{n}_t$, from the target and adding the mean of the target;

$$d_t(i) = n_t(i) - \frac{\overline{n}_t}{\overline{n}_r} f_r(i) + \overline{n}_t. \tag{4.3}$$

In general, the differential photometry will contain some more noise than the photometry of the target alone, depending on the brightness of the reference star. With $\sigma_t^2 = \mathrm{Var}(n_t)$ and $\sigma_r^2 = \mathrm{Var}(n_r)$, the total noise for differential photometry becomes

$$\sigma_d = \sqrt{\sigma_t^2 + \left(\frac{\overline{n}_t}{\overline{n}_r}\right)^2 \sigma_r^2}. \tag{4.4}$$

4.2.7 Remove the Sky

The sky level is normally computed from a separate window/channel, which should appear at the end of the window list. All the pixels in the window, except for the

first four columns, are used. The sky level is computed by first computing a N-sigma noise threshold, and rejecting values outside this range to avoid including cosmic rays and dead pixels in the sum. The number of sigmas, N, to use in the median filtering can be set by calling `rtp` with the `-m <nsig>` option

As an alternative to sky field processing a sky annulus value can be computed from the outer part of each individual channel and used for the sky level subtraction. This option has provided to be particularly useful when the sky level shows significant gradients across the field, as can be the case during a full moon, or when the sky level changes rapidly combined with strong gradients, as during Auroral conditions. Sky annulus processing can be activated by invoking `rtp` with the `-s <rad>` option. The number `<rad>` gives the number of pixels along each edge to be used. The first four columns in the image are as always avoided.

As we shall see, when these methods are used to process real data in the next chapter, sky level corrections will introduce noise into the data. There are also significant differences between the alternative approaches. First of all, using a sky field gives a lot more pixels to average over than the sky annulus method. Second, due to the nature of the differential photometry method, the sky level correction will self cancel when the reference star has the same intensity as the target, thus the noise also disappears, since the same number is used for both the n_t and n_r terms in Eq. 4.3. In other cases, the noise from the sky channel still contributes less to the total noise than the noise from the reference star channel. Writing $n_t = m_t - s$ and $n_r = m_r - s$, where m_t and m_r are the raw counts from the target and reference star, and s is the measured sky level scaled to the area of the aperture, we can rewrite Eq. 4.3 as

$$d_t(i) = m_t(i) - \frac{\overline{n}_t}{\overline{n}_r} m_r(i) - \left(1 - \frac{\overline{n}_t}{\overline{n}_r}\right) s(i) + \overline{n}_t. \tag{4.5}$$

Thus, the contribution to the noise from the sky channel is scaled by the factor $(1 - \overline{n}_t/\overline{n}_r)$, when using a sky field for the sky level correction. However, when using the sky annulus method, the sky is measured independently around the target and reference star, s_t and s_r, and the differential photometry becomes

$$d_t(i) = m_t(i) - s_t(i) - \frac{\overline{n}_t}{\overline{n}_r}(m_r(i) - s_r(i)) + \overline{n}_t. \tag{4.6}$$

In this case the noise becomes a sum of four independent variables (subject to some scaling depending on the ratio between the means of the two stars).

In detail, since the ns, and s's are independent variables, the noise in Eq. 4.5 becomes

$$\sigma_d = \sqrt{\sigma_t^2 + \left(\frac{\overline{n}_t}{\overline{n}_r}\right)^2 \sigma_r^2 + \left(1 - \frac{\overline{n}_t}{\overline{n}_r}\right)^2 \sigma_s^2}, \tag{4.7}$$

and the noise in Eq. 4.6 will be

$$\sigma_d = \sqrt{\sigma_t^2 + \sigma_{s_t}^2 + \left(\frac{\overline{n}_t}{\overline{n}_r}\right)^2 \sigma_r^2 + \left(\frac{\overline{n}_t}{\overline{n}_r}\right)^2 \sigma_{s_r}^2}, \tag{4.8}$$

```
------------------------------------------------------------------------
 RTP  -  real time processing, Version 0.92 (Roy H. stensen 2000)

Displaying:      Differential photometry
Plot type:       Full plot
Data style:      Lines
Log scale:       Off
Autoscale:       On
Window size:     50x50 pixels
Aperture radius: 14.00 pixels (area=615.75)
Aperture center: 18.25,19.83
Filter:          B
Plot range:      x = [0:1140], y = [0:0]
Plot channels:
     #1-#2:   ON      #1-#3:   ON      #1-#4:   ON
     #2-#3:   OFF     #2-#4:   OFF   #1-#ALL:   ON
Next data file: data/w00056.fits
------------------------------------------------------------------------
```

Figure 4.3: Status window seen while running the `rtp` program.

when σ_s^2 is the variance of the sky field and $\sigma_{s_t}^2$ and $\sigma_{s_r}^2$ is the variance of the sky annulus of the target and reference star respectively.

Note that since the variance is inversely proportional to the number of pixels in the sample, and the sky annulus always contain less pixels than a sky field, this further degrades the signal to noise ratio when applying the sky annulus method. Typically the sky annulus region will contain at most half the pixels of a full window, so that if the sky samples are otherwise equivalent, $\sigma_{s_r}^2 = 2\sigma_s^2$.

4.2.8 Changing the display

When `rtp` runs, it displays the propagating light curve in the GnuPlot display, and status information in the terminal window (shown in Fig. 4.3). The 'd' key switches from raw data (preprocessed data without sky subtraction), processed data (data with sky subtraction) and differential photometry, and the current mode is reflected in the `Displaying:` field on the status display. The number keys will toggle on and off what channels to display, while the corresponding `Plot channels:` field on the status display reflects the current situation.

The cursor keys will move the current view of the light curve up/down and forwards/backwards, while the Ins, Del, PgUp, PgDn, Home and End keys will change the scales of the plotted light curve.

```
------------------------------------------------------------------------

Keystroke commands:

  d  - Plot Raw or Processed data        p  - Plot full or ranged plot
1-9  - Plot channel # toggle             s  - Plot sky channel toggle
  b  - Plot bias level toggle
  P  - Make PostScript plot (rtplot.ps)  L  - Make LaTeX plot (rtplot.tex)
 +/- - Bigger or Smaller aperture
 []  - Move aperture left or right      {}  - Move aperture down or up
  M  - Make new Mask (mask.fits)         R  - Reprocess all data
  T  - Test image (testview.fits)
  ?  - Print Help                        q  - Quit

------------------------------------------------------------------------
```

Figure 4.4: Help page for the `rtp` program.

4.2.9 Producing plots

It is possible to save the currently displayed plot as a postscript file with the 'P' key. The filename will be `rtplot.ps`. To get the name of the target object on top of the plots, use the 'O' keystroke command, and enter a string object identification.

4.2.10 More commands

Use the '?' key to display the list of keystroke commands while running `rtp`. This will produce a help page that looks something like what is shown in Fig. 4.4.

4.2.11 Command line options

A number of command line options are available to control the behavior of `rtp` as it is started. By convention, lower case option flags take an argument while uppercase does not. Table 4.1 shows the complete list of options and their effects.

4.2.12 Output data

`rtp` uses circular apertures with an arbitrary center and radius within the aperture windows when computing the total flux inside the windows, as described above. Sky values are computed with median filtering to avoid cosmic rays, and are derived

Table 4.1: `rtp` command line options.

`-d <sec>`	Delay between the processing of each frame. Default is 0.
`-f <name>`	Name of flat fieldimages. Default is `flat*.fits`.
`-l <name>`	Name of logfile. Default is `rtp.log`.
`-m <nsig>`	In sky level processing, exclude pixels that deviate more than `<nsig>` sigma around mean value. Default is 3.0.
`-n <num>`	Start processing at number `<num>`. Default is 0.
`-o <name>`	Name of the target star.
`-r <rad>`	Aperture radius [pixels]. Default is the maximum that will fit in the window, excluding the first four (bad) columns.
`-s <size>`	Size of sky annulus [pixels]. Defaults to 4.
`-t <num>`	Which sky field to use for sky level processing. Defaults to 1.
`-x <pos>`	Aperture x-position. Defaults to the centre, excluding the first four (bad) columns.
`-y <pos>`	Aperture y-position. Defaults to the centre.
`-z <num>`	Number of sky fields. Defaults to 1.
`-B`	Non-interactive (batch mode) processing; generates output data files and terminates.
`-C`	Compute and save centering parameters.
`-D`	Display windows images in `saoimage` as they arrive.
`-M`	Moving aperture photometry.
`-O`	Compute individual centering offsets for each star.

from all good pixels in the window and scaled to the area of the current processing aperture. These sums are continuously appended to the file `phot.raw`, as soon as each new integration has been received and stored by `tcpcom`.

Sky subtracted photometry values are computed, either by using the value from the last ("sky channel") aperture or by using a sky annulus around each aperture, if the windows were set up large enough to allow this. Sky subtracted photometry for each star is appended to the file `phot.dat`.

Differential photometry is computed for the target (channel #1) and each reference star. If more than two reference stars are available, differential photometry is also computed between the second star (channel #2) and the remaining channels so that any periodic variability in the reference stars can be checked for easily. If possible a differential photometry value is also given for the target relative to the sum of all reference stars, which is useful in fields where all reference stars are significantly fainter than the target. These values are continuously appended to the file `phot.dif`.

While waiting for new data to arrive, `rtp` calls `gnuplot` to show plots of the photometry files so that the observer can investigate the light curves.

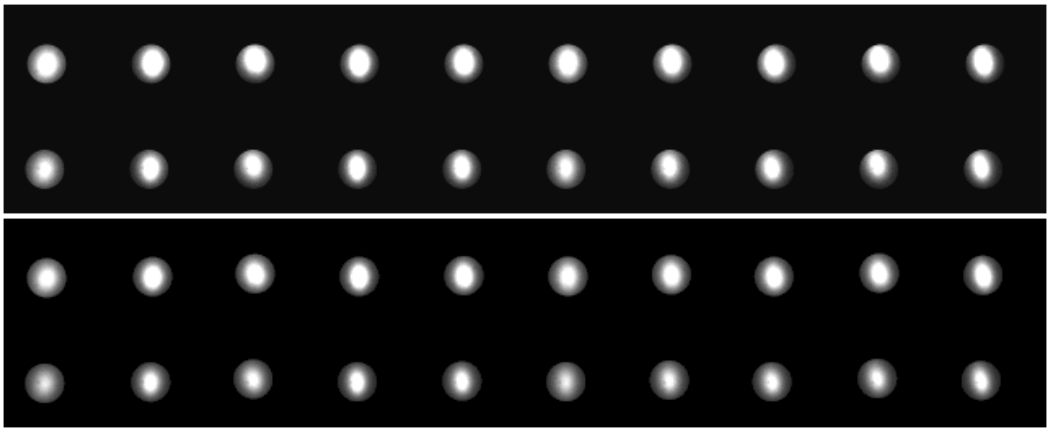


Figure 4.5: Example of a short sequence of images as viewed by `rtcnv`. The two stars are target at bottom and reference star on top, and time runs from left to right. The upper panels have been processed with constant offsets, and the lower panel with the moving aperture (MAP) technique.

4.2.13 A better view

True to the C++ object-oriented philosophy the actual processing in `rtp` was implemented as a class library, `RTP`, with the main program only concerned with handling the input/output. Another program, called `rtcnv`, uses the `RTP` class to process the same images, but instead of computing aperture sums for output, it converts the requested window frames into a single FITS image to give the observer a direct view of what is going on in the field.

When `rtcnv` is started, the desired number of window frames to convert, the number of the first frame and the offset between following frames are given, together with any `rtp` processing options. The two strips in Fig. 4.5 have been produced by `rtcnv`, with and without the MAP option.

The options that can be used with `rtcnv` are the same as those of `rtp` listed in Table 4.1, with some additional options to control what frames are included in the resulting FITS image. The additional options are included in Table 4.2, below.

4.3 Real time Fourier Transforms

An adapted version of the Quilt program `qsft` (see Ch. 1.2.5), called `rtft` can be used to make Fourier Transforms of the photometry files as the observations are being made. This gives the observer the opportunity to inspect the data in real time, and for search programs trying to detect new variable stars this option has turned out to be very useful [OEs00b].

Table 4.2: `rtcnv` command line options.

`-n <num>`	Number of sequence frame to convert. Default is one frame only.
`-o <num>`	Offset between sequence frames. Default to follow the sequence, *i.e.* an one frame offset between frames.
`-s <num>`	Start processing at frame `<num>`. Default is zero, *i.e.* the first frame in the sequence.
`-w <num>`	Number of windows to use. Default is to use target, all reference fields and all sky fields, and not the bias window.
`-B`	Don't remove bias level.
`-R`	Raw sequence (no conversion). Invoking this options produces the windows sorted in the CCD readout direction, and not with target at bottom, followed by reference stars and sky fields.
`-S`	Don't subtract sky values.
`-X`	Don't use the aperture mask.

Table 4.3: `rtft` command line options.

`-a`	Compute amplitude spectrum (default).
`-P`	Compute power spectrum.
`-p`	Also write a peaks file.
`-N`	Do not normalise the data.
`-n <nsft>`	Pre-set the number of frequency samples.
`-o <file>`	Write spectrum to `<file>`.
`-f <filt>`	Get filter number `<filt>`.
`-F <nfilt>`	Number of filters in data file.

`rtft` is called with one or three parameters, in the same way as `qsft`. The last parameter is always the name of the file to take the FT of, and the first and second optional parameters give the desired frequency range. In addition a number of option flags are understood, and all listed in Table 4.3.

The `-N` option can be used to prevent `rtft` from normalising the data. Normalisation will otherwise be performed unless `rtft` detects the line "`# rtcorr: data have been normalised`" in the input data. Unlike `qsft`, `rtft` will probably not work well with data from multiple runs, as this part of the `qsft` code has not been tested with the modified input. This question must be resolved once we have a reliable way to get precision timing into the CCD control system. For the moment, the only changes that have been done to the `qsft` code when putting `rtft` together, except for the porting to C++, is related to the input of data from `rtp`, and the format of

the output files. The `rtft` program still needs to be worked on, but it does the job of providing the observer with instant Fourier transforms of the real-time photometry, as intended.

The output from `rtft` is put into a file which is given the `.sft` extension, and has the same column format as the output of `rtp`, except that the first column contains frequency instead of time. The peaks files are generated one for each column in the input file, with the extensions `.pkN` where `N` is the column number.

4.4 Extinction corrections

Extinction correction has not been implemented as a default part of `rtp` yet. This is due to the fact that such corrections requires information about the zenith angle of the target as the observations proceed. Although not difficult to compute, it requires at least knowledge about the coordinates of the target and geographical location of the observatory. Since this information is not yet conveyed from the telescope system to the instrument computer (as it certainly should), the observer will have to provide this information manually.

A library has been written to handle the necessary astronomical calculations to provide extinction corrections (class: `Obsinfo`). To do extinction corrections it is necessary first to know the position of the target, date and time of observation and the location of the observatory. Since this information currently is not available via the FITS headers, it must be provided to the program by the user. To avoid the painstaking labor of typing in all this information each time the program is run, input is provided through two simple text files. The first one, named `TELESCOPE`, contains information about the observatory and instrument, and will be the same throughout an observation run. The second one, named `TARGET`, contains the information that changes from target to target. Below is an example of a `TELESCOPE` file.

```
Observatory:    "Nordic Optical Telescope"
Longitude:      "-17d53m06.3s"  # Negative for West / Positive for East
Latitude:       "+28d45m26.2s"
Altitude:       "2465.5m"
Instrument:     "HiRAC"
Filters:        "-UBVRIabX"     # Filter in position 012345678
```

The format is quite intuitive. The only absolutely vital field here is the `Longitude:` field. The other fields are used to add information to the header part of all output files. Note that the "dms" separators in the longitude and latitude fields are mandatory to identify the field as given in degrees, minutes and seconds. Actually, "hms" is also allowed, and will invoke conversion from hour angle to to decimal degrees.

In the future this file should be generated by `tcpcom` automatically using information from the telescope information system. The `Filters:` field is used to convert filter numbers in the FITS headers to their equivalent filter labels, and must be set to match the order of filters that were fitted in the filter wheel at the time of observations. Since the filter unit in question is the one on the HiRAC camera, it has 8 positions starting with number 1. The zero position is therefore marked with a `-`. `a` and `b` are narrow band filters (H_α and H_β), and `X` indicates no filter. For the Tromsø CCD Photometer the sequence would currently look like `"BVRI+X"`, where the `+` indicates the block position. Note that comments starting with a `#` character can appear on any lines in these files.

The `TARGET` file can look something like this:

```
# Target information:
TARGET:          "HS 0600+6602"
RA(B1950):       "06h00m21.7s"
DEC(B1950):      "+66d02m31s"
# Observation run information:
FILTER:          "B"
DATE:            "1999-10-17"
TIME(UT):        "05:20:26"
# Extinction coefficients:
K_U:             "0.49"
K_B:             "0.30"
K_V:             "0.15"
K_R:             "0.09"
K_I:             "0.05"
```

Obviously, the target coordinates are essential. They can be given either in epoch 1950 or 2000 coordinates, so the parenthesis in the RA and DEC fields must contain either 'B1950' or 'J2000'. These coordinates will be automatically precessed by `Obsinfo` to the date given in the `DATE:` field. The `TIME(UT):` field is also mandatory, and is used as the starting point of the observation series. The correct exposure interval is computed from the difference between the first data points in the input data file. If the time stamps on the following data points does not proceed with this interval, an error message appears.

The `FILTER:` or `FILTERS:` field, contain one or more filters used in the observation series. If more than one filter is given, `Obsinfo` will assume that the filters given are used in sequence throughout the series, with the first filter corresponding to the first data point, the next to the second, and so on in a loop to the last data point. During extinction correction, the procedure `Obsinfo::ext_corr()` applies the extinction coefficient corresponding to the filter used for that observation, in the sequence given by the `FILTERS:` keyword. For the moment, the only filters that can be associated with an extinction coefficient in this way are UBVRI, specified by

`K_U, K_B` *&c.* The exact extinction coefficients must be calculated manually for each night of observations, but the values given in the example `TARGET` field are useful estimates for observations at the NOT.

`rtcorr` is called with the name of the file to process as input. This would normally be the `.dat` file, for instance the default `phot.dat`. The raw data input file (the one with the `.raw` extension) can also be given as input, in which case `rtcorr` will perform the sky subtraction, and produce another data file, this time with the `.ras` extension with the result of the sky subtracted photometry. This file would normally be equivalent to the `.dat` file produced by `rtp`. But `rtcorr` has one option that `rtp` cannot replicate; sky level smoothing. Calling `rtcorr` with the `-b <size>` option, will apply a moving average `<size>` pixels wide to the sky channel or the sky annulus data, before doing the sky level correction.

Only two more modifiers are understood by `rtcorr`. The `-D` option will cause `rtcorr` to produce lots of debug information while it process the data, in particular the computed airmass for each step. The other option is `-w <w>` where `<w>` is the percentage of the light curve to apply the *tapering* window function to. `<w>` defaults to 10, which will taper the first and last 10% of the light curve. More details on the tapering procedure is given below.

Extinction correction is performed on the sky subtracted data by multiplying each measurement in each channel with a coefficient that depends on the extinction coefficient for the filter in question k_f and the airmass $A(t)$,

$$C = 10^{0.4 k_f A(t)}. \tag{4.9}$$

The extinction corrected data is stored in a file with the same base name as the input file (*e.g.* `phot.dat`), and with the extension changed to `.ext` (*i.e.* `phot.ext`). The next step that `rtcorr` does is to call the `Obsinfo` function `diff_phot()` to recompute the differential photometry from the extinction corrected photometry. This is then saved to a file with the `.edf` extension (*i.e.* `phot.edf`).

The extinction corrected differential photometry is next normalised by fitting a straight line through the data using a least squares algorithm, subtracting this line and dividing by the average of the data to produce normalised modulation magnitudes. The corrections are done independently for each star and each filter. The resulting light curve is stored in a file with the `.nma` extension (*e.g.* `phot.nma`).

The final step that `rtcorr` performs is to apply the tapering window function to the normalised data. By default, the first and last 10 % of the data are tapered, so that the light curve smoothly starts off from and drops off to zero. The result is then saved to a file with a `.win` extension. The window function applied here is the so-called "Welch window" [Pre92, p. 554],

$$w_j = 1 - \left(\frac{j - \frac{1}{2}N}{\frac{1}{2}N} \right)^2. \tag{4.10}$$

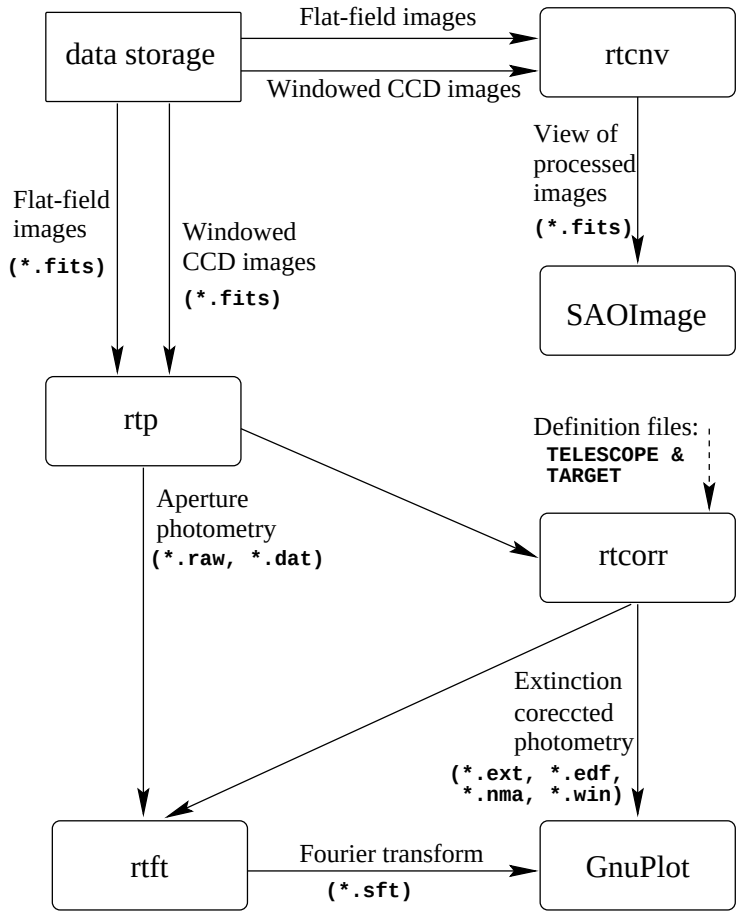


Figure 4.6: Schematic of the interactions between the different programs when processing stored windowed photometry data.

(Note that the names for these files need not start with `phot`. Any name base provided for the original input file will be inherited by the produced files.) All these files are acceptable as input to `rtft`, which will perform the necessary normalization if this has not already been done, but `rtft` does not do any windowing by itself. The observer must decide whether or not windowing is appropriate for the data set in question. In most cases using a window will reduce the sidelobes of any peaks in the amplitude spectrum, but inevitably some amplitude is lost from the peaks. A thorough investigation of the effects of windowing on the resulting amplitudes has not been attempted.

4.5 As it Is

A schematic view of the interactions between the different programs described in this chapter for reprocessing of the data is shown in Fig. 4.6. The intention with these

programs has primarily been to produce quick and reliable aperture photometry that can be investigated in real time. Some additional tools have been developed out of necessity to do the basic calibrations required to get accurate photometry. As will be evident in the next chapter, when we put these programs to the test on some real photometric data, the signal to noise in the final amplitude spectrum is very much dependent on the options the user chooses when reprocessing the data. Figuring out which out of the many options available to invoke in order to get the best possible result, therefore becomes quite an art. The methods and options available in the programs presented here is by no means exhaustive, and many cases can be imagined where the observer can improve the results significantly by careful analysis of the output data and implementation of new methods. For this reason, all the resulting output aperture photometry data are stored as plain text files, so that it will be easy to import the data into other programs for further analysis and processing.

Chapter 5

Put it to the Test

First light for the Tromsø CCD Photometer came on the 1^{st} of May 1999, during two nights of technical time at the Nordic Optical Telescope (NOT). The target selected for the test was PG 1336−018, a variable subdwarf B star, that had been one of the targets of the just completed XCOV 17 campaign of the Whole Earth Telescope (WET). The test run was very successful, and our data were contributed to the campaign to extend the coverage. Before we give the details of the test run and show the results, let us give a short briefing on the particularities of the recently discovered pulsating subdwarf stars.

5.1 The Rhythm of the Subdwarfs

The hot subdwarf B (sdB) stars are evolved low mass objects ($\sim$ 0.5 $M_\odot$) with helium burning cores, and are surrounded by a hydrogen surface too thin to sustain H-shell burning. Their high temperature and brightness place them close to the extreme horizontal branch (EHB) in the HR diagram. One expects that these objects have been subject to a core helium flash and some substantial mass loss during or after the red giant phase. After core helium exhaustion they are expected not to evolve toward the asymptotic giant branch (AGB), but instead move left in the HR diagram, passing through the subdwarf O population, and ultimately enter the white dwarf population as low mass WDs [Sil00b]. A long standing mystery, known as the horizontal branch problem, is how red giants can loose such a considerable fraction of their H-rich envelope after the ignition of core helium. The hot subdwarfs display this problem in an extreme fashion. The discovery that some of these objects pulsate open up the possibility for probing the interior of these stars with the tools of asteroseismology, and can be the key to a better theoretical understanding of the processes in these stars, and may one day provide the answer to the horizontal branch problem.

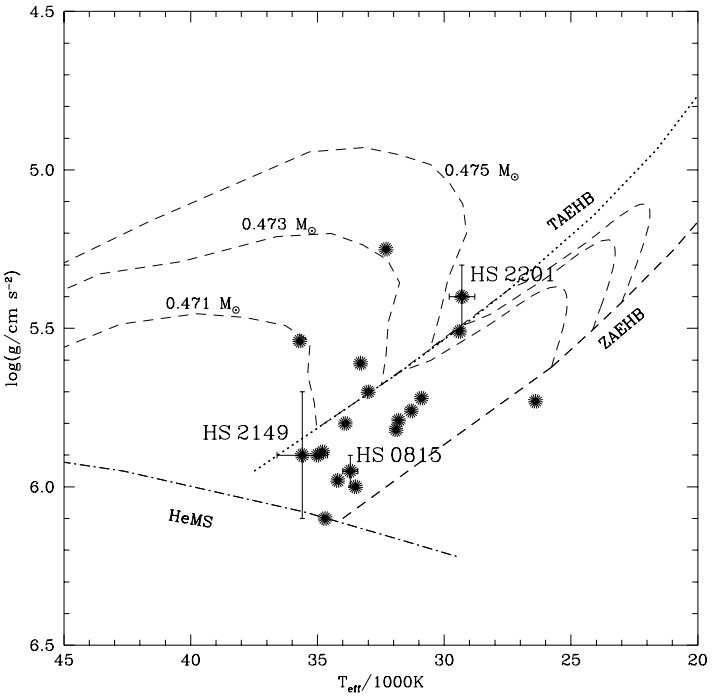


Figure 5.1: Comparison of the three newly discovered sdBVs to previously known ones in the ($T_{\rm eff}$, log g) plane. The position of the zero age extreme Horizontal Branch (ZAEHB), the terminal age extreme Horizontal Branch (TAEHB), the helium main sequence and evolutionary tracks for EHB stars [Dor95] are also shown.

This possibility was originally proposed by Charpinet *et al.* [Cha96] as they presented their theoretical prediction for pulsations in the sdB stars. Simultaneously, the South-African group working with the Edinburgh-Cape survey had discovered several cases of such pulsations in sdBs, the first, EC 14026−2647, which was reported by Kilkenny *et al.* in 1997 [Kil97]. The South-African group has published data on a total of 13 variables from their sample, as reported by O'Donoghue *et al.* [ODo99]. These pulsating sdB stars are sometimes referred to as EC 14026 stars, after the prototype, or just sdB variables – sdBVs for short.

A particularly bright sdBV was reported by our group at the meeting of the Whole Earth Telescope at Bonas, France, in August 1999, after being discovered at our first sdBV search run at the NOT in July [Sil00a]. Another sdBV, PG 0856+121 was recently discovered by Picconi *et al.* [Pic00], and Billères *et al.* has also presented one more this year [Bil00]. Three more sdBV stars, discovered by us in October 1999 and presented in the next chapter, brings the total number up to 19. We have also two found two more from our observation run of July 2000, but the results from these are not yet ready for publication.

Table 5.1: Known sdBV stars, vital parameters and references.

Name	m_V	$T_{\rm eff}$	log g	Co	P_+	P_-	A	N_p	Ref
EC 14026−2647	15.3	34 700	6.10	G2	144	134	12	2	[Kil97]
PB 8783	12.3	35 700	5.54	F3	136	94	9	7	[Koe97, ODo98a]
EC 10228−0905	15.9	33 500	6.00	G0	152	140	14	3	[Sto97]
EC 20117−4014	12.5	34 800	5.89	F6	158	137	4	3	[ODo97]
PG 1047+003	13.5	35 000	5.90	?	162	104	10	13	[Bil97, ODo98b]
PG 1605+072	12.8	32 100	5.25	?	529	206	64	50	[Koe98a, Kil99]
PG 1336−018	13.5	33 000	5.70	M5	184	141	10	2	[Kil98, Ree00]
KPD 2109+4401	13.4	31 200	5.84	?	198	182	9	5	[Bil98, Koe98b]
Feige 48	13.5	28 900	5.45	?	379	343	6	4	[Koe98c]
PG 0911+456	14.6	31 900	5.82	?	166	156	6	3	[Koe99a]
PG 1219+534	13.2	32 800	5.76	?	149	128	9	4	[Koe99a]
EC 05217−3914	15.6	31 300	5.76	?	218	215	5	4	[Koe99b]
KUV 0442+1416	15.1	33 000	5.72	?	232	185	20	7	[Koe99b]
PG 0856+121	13.5	26 400	5.73	?	435	313	4	2	[Pic00]
KPD 1930+2752	13.8	33 300	5.61	?	145	332	5	44	[Bil00]
PG 1618+563B	13.5	33 900	5.80	F3	144	139	2	2	[Sil00a, Sil00b]
HS 0815+4243	16.1	33 700	5.95	?	131	126	7	1	[OEs00b]
HS 2149+0847	16.5	35 600	5.99	?	159	142	11	2	[OEs00b]
HS 2201+2610	13.6	29 300	5.40	?	350	350	10	1	[OEs00b]

A list of these sdBVs are found in Table 5.1. The table gives magnitudes, effective temperetures and gravities from spectroscopic observations, companion spectral class (if detected), the longest and shortest periods (P_+ & P_-) in seconds, as well as the amplitude of the strongest period (A) in milli modulation amplitudes (mma), and the total number of periods seen (N_P). Fig. 5.1 shows the locations of these new variables relative to previously discovered variables in the $T_{\rm eff}$/log g plane (data from Table 5.1).

All the pulsating sdB stars found so far have effective temperatures in excess of 25 000 K and log g above 5.0. From Table 5.1 we see that their periods range from 90 to 530 seconds, and with the number of modes ranging from one to as much as fifty. This range of parameters fit nicely with the predictions of Charpinet *et al.* [Cha96], and is further pinpointed in their article presenting a driving mechanism for the pulsations [Cha97]. The idea is that the pulsations are driven by an opacity bump caused by a local enhancement of the iron abundance in the envelope of these stars.

Our target of occasion, PG 1336−018, discovered to be a sdBV by Kilkenny *et al.* [Kil98], but with the extra peculiarity that it is also an eclipsing short period binary, giving it a surprisingly spectacular light curve. The companion is an M type dwarf, and the system has a binary period of 2.4 hours and an inclination angle of about 81°. With these parameters one can expect that PG 1336−018 is rotationally

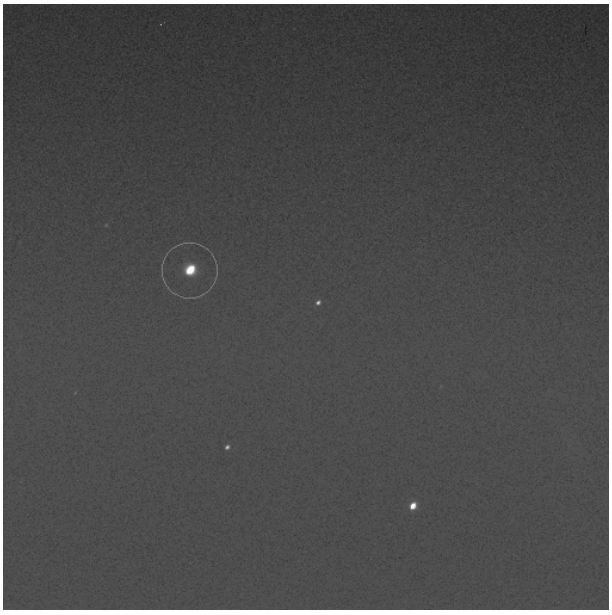


Figure 5.2: PG 1336−018 field image. The target is the brightest star (circled) and the second brightest was used as reference star. The CCD's limited field of view did not allow any brighter star to be used as reference star.

locked to its companion, thereby giving it a known rotational velocity. Thus, this star was expected to be able to reveal some of the details necessary to understand the pulsation spectrum of the sdBV stars, and was selected as the primary target for the campaign [Ree00].

5.2 A Night at the Dome

Our CCD camera head was mounted on the filter and shutter unit of the High Resolution Adaptive Camera, replacing the standard liquid Nitrogen cooled CCD instrument, since our own FASU was not completed at this time. The CCD camera was fitted with an engineering grade Tektronix TK1024A CCD chip on loan from the CUO. The HiRAC is equipped with an additional small 64×64 frame transfer high speed CCD camera that controls the adaptive optics for the main camera. We did not run the instrument in its adaptive optics mode, spatial resolution not being a priority, so the tip-tilt controlling camera was kept off-line, and we used the water cooling connection for this to cool our own camera. The CUO controller box for our camera were mounted in the place of the similar controller for the HiRAC's science instrument, and programmed for the Tektronix chip using the `download` program.

During daytime a few days before our run started, the instrument was tested off-line using an artificial light source, to make sure that our interface to the HiRAC filter

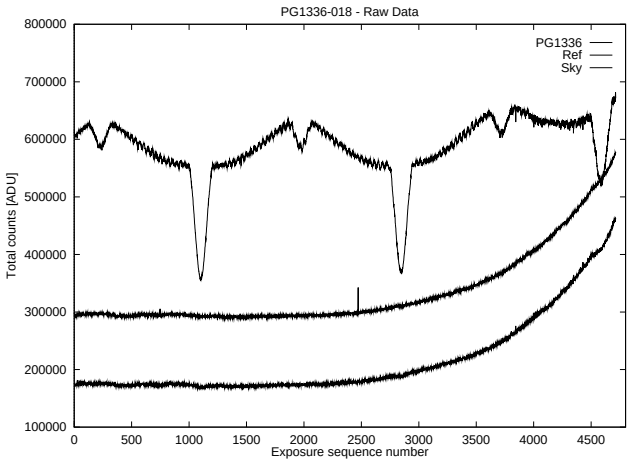


Figure 5.3: Real time light curve of PG 1336−018. The two lower curves are the reference star (starting at about 300 kADU) and the sky level (starting at about 180 kADU).

and shutter unit worked as intended. We were not able to get telescope position information back from the telescope control computer to put in the image data headers as intended, due to some problems with the software interface libraries. I suspected that some simple C/C++ conflicts was to blame, but did not have enough time to resolve this problem. All other things worked as planned, and we were quickly ready to test our system on our selected target.

5.2.1 Observations of PG 1336−018

Our first night was lost to clouds, but on the 2$^{\text{nd}}$ of May we had good conditions and started a sequence of 3 second integrations with a sequence interval of 5 seconds without any filter, as were required for the WET campaign. After 6.5 hours the target was setting and we ended the run after obtaining a total of 4717 sequential integrations. Fig. 5.3 shows the real-time light curve of the raw data, and as is obvious from the figure, the sky level is very high due to the full moon, and increases rapidly as the airmass increases towards the end of the night.

Fig. 5.2 is a full frame 3 second unfiltered integration of PG 1336−018. As can be seen in the image, no reference stars with similar brightness to PG 1336−018 are present. The reference star used, the second brightest in the frame, is 1.5 magnitudes fainter than PG 1336−018, and gives adequate results.

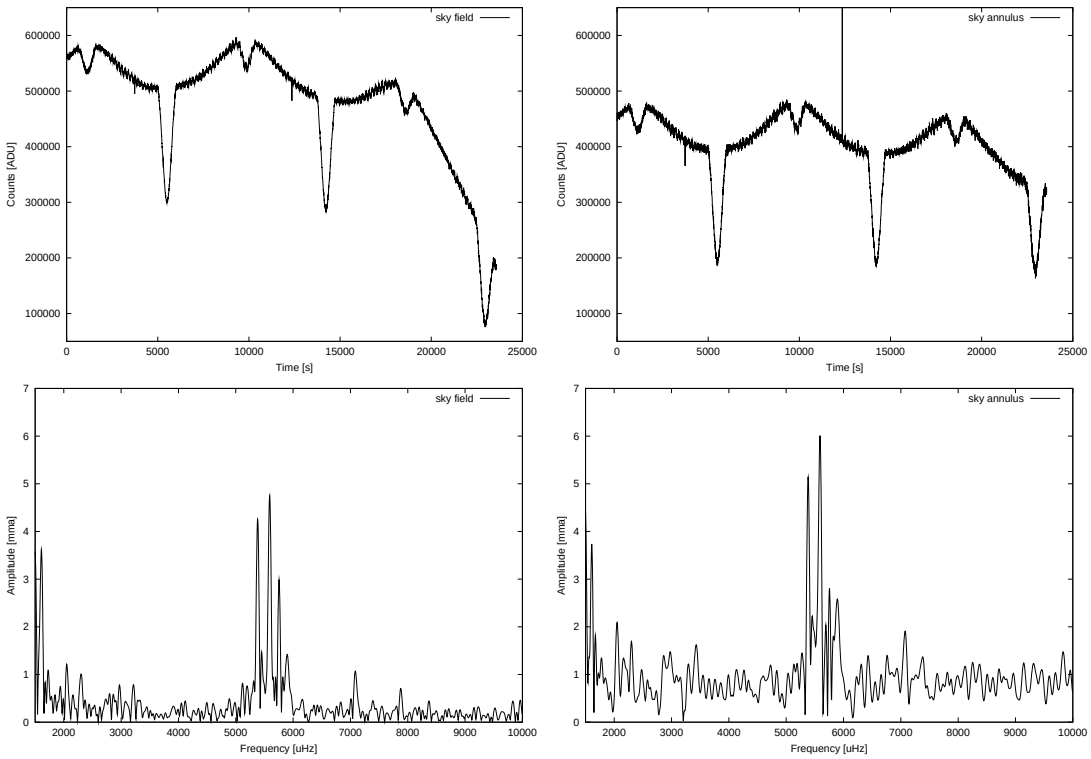


Figure 5.4: PG 1336−018 light curves (upper panels) and amplitude spectra (lower panels), generated using the sky field (left) and sky annulus (right).

5.3 The Deeper Picture

At the time of the test run, only the raw and sky subtracted data were available from the real time photometry program, but the Fourier transform of the sky subtracted data still showed in a very impressive way that our system worked to the specifications. However, the full moon did give us some unexpected problems when analysing the data. A strong gradient was present over the field throughout the observations. At the beginning of the observations the sky field indicates a sky count of about 73 kADU when using an aperture with a radius of 20 pixels. But when measuring the counts along the edge of the target and reference star windows, we find sky levels corresponding to 113 kADU and 121 kADU, respectively. Comparing this to the total counts in the target and reference star aperture, 542 and 226 kADU respectively, we understand that subtracting sky level using the sky field directly will significantly underestimate the sky level, and therefore the sky subtracted aperture sums will be to high.

The sky annulus mode was implemented specifically to deal with the background gradient problem in these runs, and significantly improved the shape of the sky subtracted light curve, as can be seen in Fig 5.4. It is obvious from these figures that the sky annulus does the best job correcting the overall shape of the light

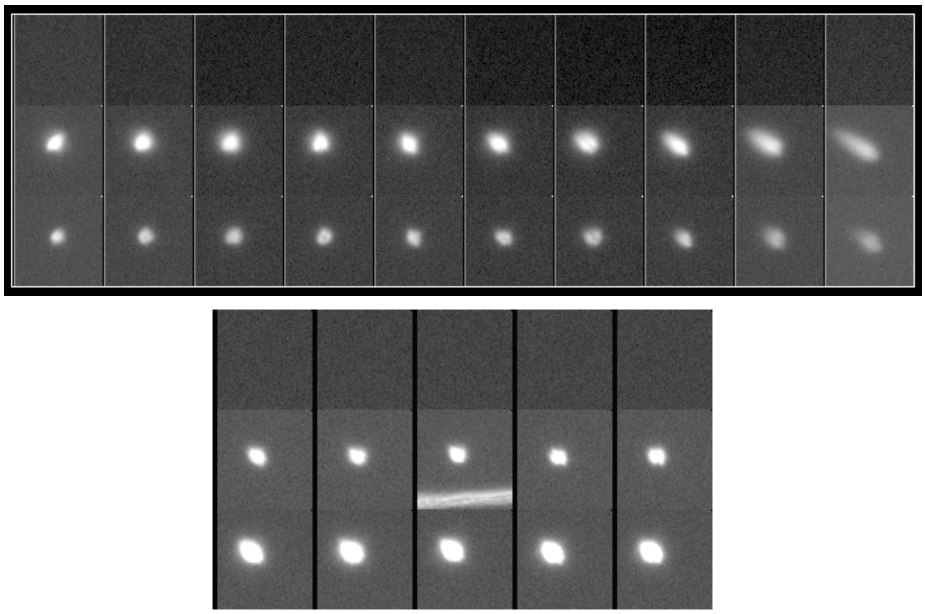


Figure 5.5: View of selected apertures from PG 1336−018 run. The upper panel shows selected frames from the beginning to the end, and the lower panel shows frames number 2472 to 2476, where a meteor has entered the picture. In the upper panel the target is in the middle, while in the lower panel the target is at the bottom.

curve in this case, but the noise level in the amplitude spectra shows a significant degradation. In fact, the noise level in the amplitude spectrum of the photometry using the sky annulus method is much worse than using the raw data directly, although this is entirely due to the extremely stable conditions during this run – any cirrus activity would have had severe impact on the amplitude spectrum of the target data alone. In the following discussion we will assume that differential photometry is mandatory, as it will be for most observation runs.

5.3.1 View Through the Window

From our studies of these data, we have found that having the window images available for later reprocessing to be an immense advantage. Although we initially considered making the system in such a way that we could throw away all the large CCD data files and just keep the resulting light curves, we have now ruled out this option. Explanations for just about all deviations in the light curves can be found by inspecting the individual images. Peaks like the one in the raw light curve of the reference star at exposure number 2474 (see Fig. 5.3), and also in the sky annulus corrected differential photometry shown in Fig. 5.4 (upper right panel) at the same point ($2474 \cdot 5 = 12370$ s) can easily be investigated by looking at the original frames with the `rtcnv` program. The lower panel of Fig. 5.5 shows the frames from 2472

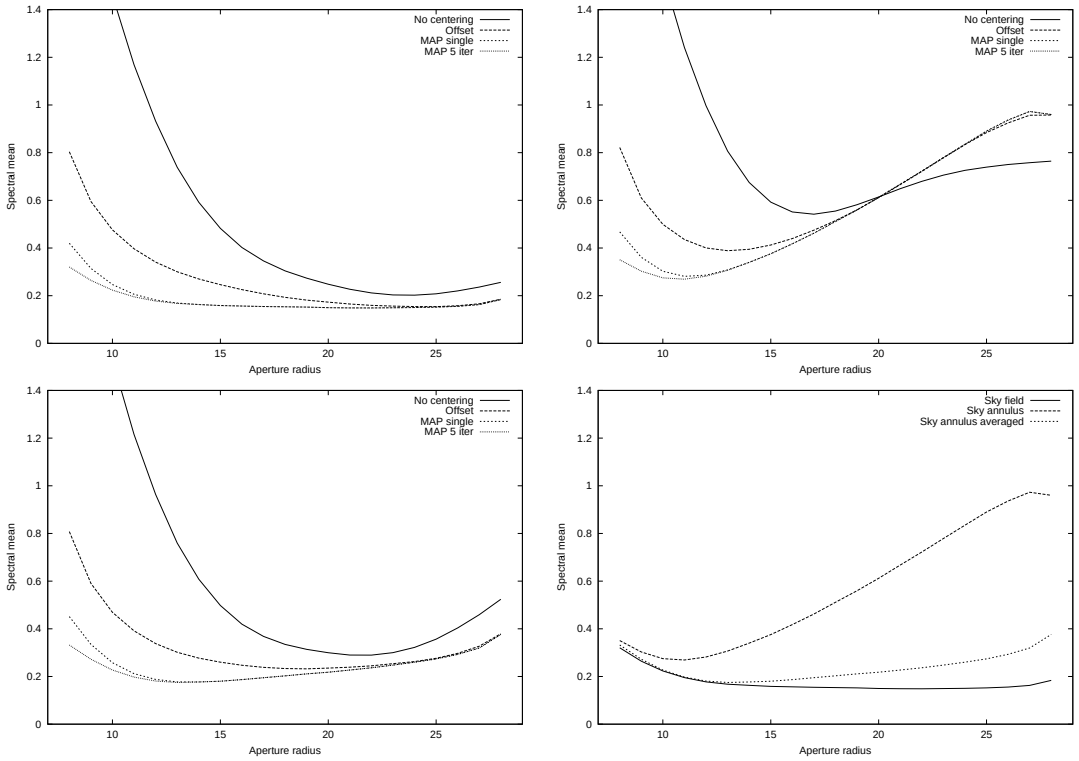


Figure 5.6: PG 1336−018 noise as a function of radius. Upper left panel shows results when using the sky field and four different centering methods. The upper right panel shows the results when using the sky annulus and the same four centering methods. The lower left panel shows the same results when using a box averaged sky annulus value. The lower right panel summarises the results for the the three different sky estimators, but with the iterative MAP centering method only.

to 2476, and clearly reveals that the culprit is a stray meteor!

In the upper panel of Fig. 5.5, the window images for ten selected frames are shown. The first corresponds to the first sequence integration, and then in steps of 500, up to exposure number 4500, almost at the end of the run. On top is the sky field, PG 1336−018 is in the middle, and the fainter red star used as a reference star is at the bottom. We can clearly see how the colour difference between the hot blue target star and the cold red reference star, shows up through different refraction by the atmosphere, as the zenith angle increases. Each window is 64×64 pixels, and the pixel size is 0.176”/pixel, giving a maximum aperture diameter that can be fitted inside such a window of about 11”.

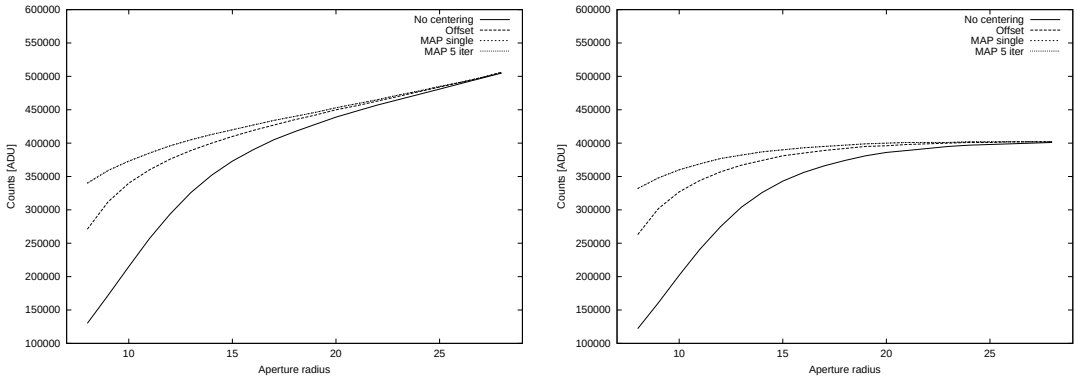


Figure 5.7: PG 1336−018 mean measured signal peak as a function of radius. The four different panels show the results of processing with the same three sky level estimator and the same four centering procedures as for Fig. 5.6.

5.3.2 A Look at the Noise

To determine how different processing methods affect the results we have processed the data with aperture radii ranging from 8 to 28 pixels, first while using the sky field and second using the sky annulus method. The results are plotted in the upper panels of Fig. 5.6, with the sky field method to the left and the sky annulus method to the right. The noise estimator used here is the mean of the amplitude spectrum ranging from the end of the area with the main pulsations at 6000 μHz and up to 20000 μHz.

Four curves are shown in each of these panels. These represent four selected centering methods used in the `rtp` processing. The solid line represents no centering, *i.e.* assume that the star is perfectly centered in the window, and remain so throughout the run. The curve with long dashes, labeled "Offset" represents the `-O` option, which centers the aperture based on the image in the first frame. The curve with short dashes represents the MAP algorithm (`-M` option), aligning the aperture on the geometric center in each frame. The dotted curve shows the results when using the MAP algorithm with iterative centering (`rtp -M -i 5`), using five iterations. In all cases, there is significant difference between the three first methods, showing the importance of good centering, especially when the aperture is small. Iterative recentering does not give much improvement here, proving that the autoguider at NOT does its job very well, since only long term centering effects are present. If there were any rapid shifts of the stars in the frame (from one frame to another), iterative recentering would have proved its value.

The difference between the panels are striking. While the sky field data appears to give good results all the way, the sky annulus data get increasingly worse as the aperture radius increases. The problem with the sky annulus data is caused by introduction of noise from the sky annulus region. Since the sky level is very

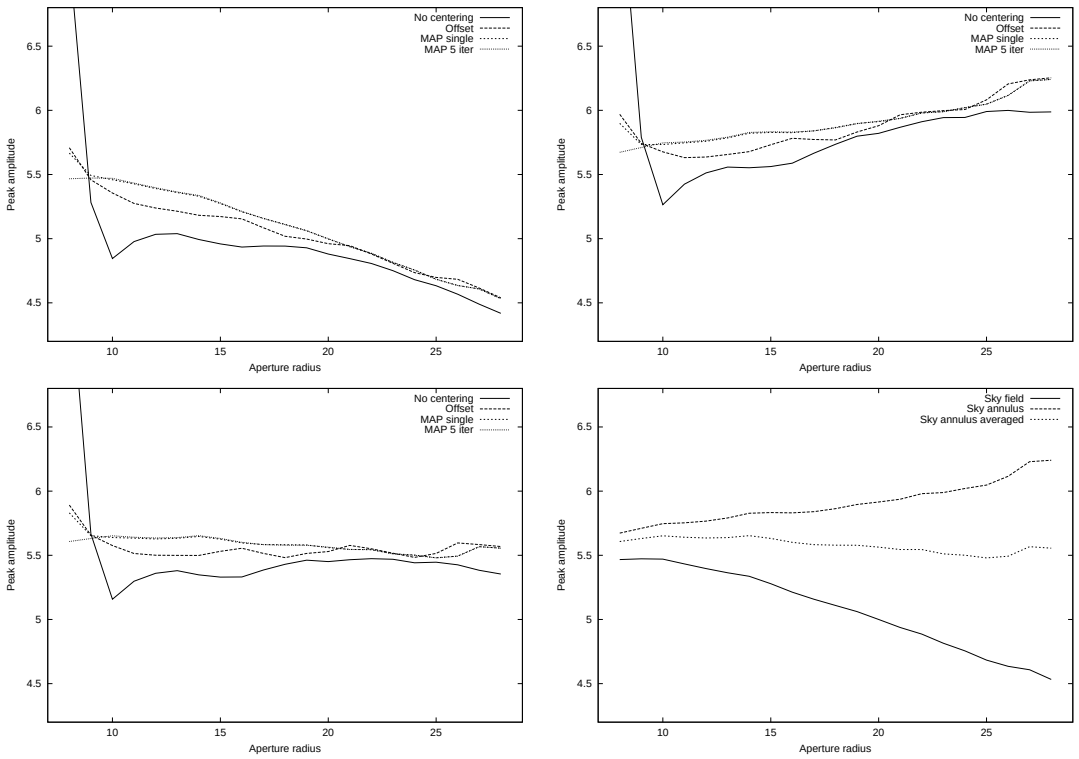


Figure 5.8: PG 1336−018 amplitude of the main peak as a function of radius. Panel order is again the same as for Fig. 5.6.

stable on short time scales in these observations, it is possible to reduce the noise from the sky subtraction by averaging over many points. In the lower left panel of Fig. 5.6 the sky annulus values have been smoothed with a 50 point wide moving average, before subtraction from each channel, giving a significant improvement. To understand why sky annulus values gives so much poorer values than sky field values, it is important to recall that sky values are computed separately for each channel resulting in a total noise that is the sum of four independent variables, instead of three. That the noise increases when the radius goes above a certain value is expected, since at this point there is no more light from the star, and only the fluctuations around zero in the sky subtracted aperture values are added into the aperture sum.

To understand why the sky field corrected photometry noise remains flat as the radius increases, recall that the sky values from the sky field gives severely underestimated values, due to the moonlight gradient over the field. Therefore we are adding more signal, not from the star but from the sky, as the radius increases, and the noise remains constant. That this is the case is seen when plotting the average aperture sum for the whole run as a function of aperture radius, as have been done in Fig. 5.7.

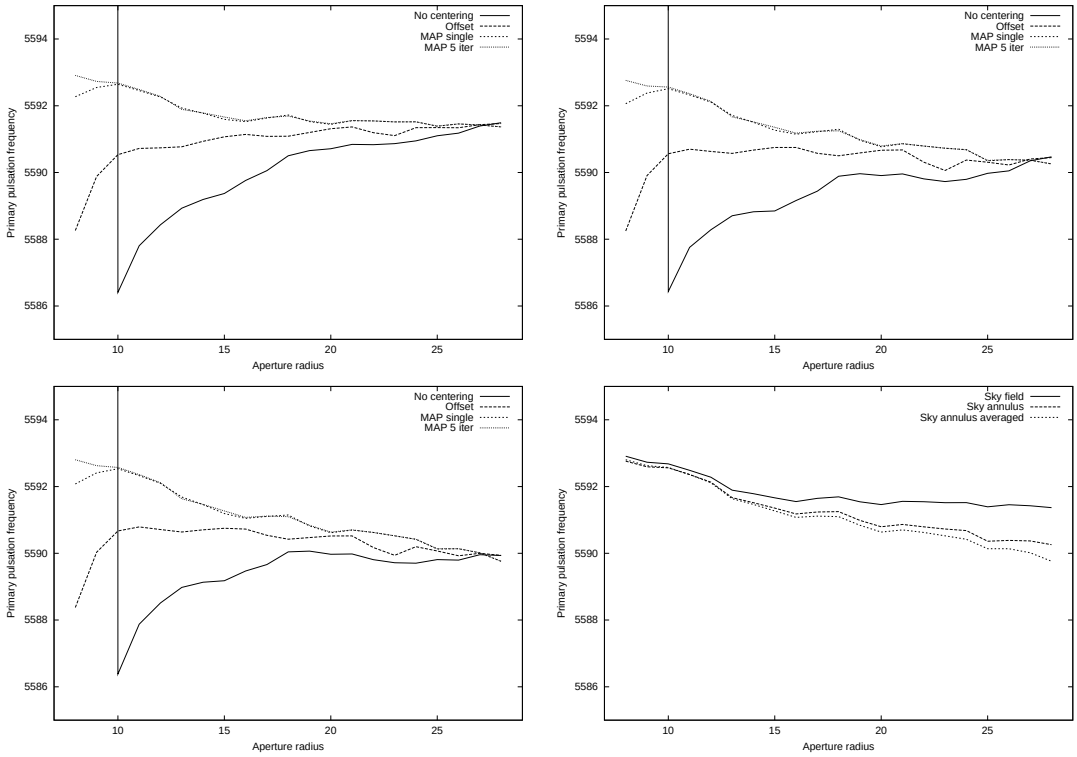


Figure 5.9: PG 1336−018 frequency of the main peak as a function of radius. Panel order is again the same as for Fig. 5.6.

5.3.3 The Highest Peak

To get an accurate estimate of the signal to noise ratio in the amplitude spectrum, we can use the measured value of the main pulsation period (at 5585 μHz) as our signal. Fig. 5.8 shows the amplitude of this peak as a function of radius for the three different methods. Again the results from the sky field correction is shown in the upper left panel, the sky annulus method in the upper right, the box averaged annulus method in the lower left, and a summary comparing the results between the three methods (using only the results from the iterated MAP centering method) is shown in the lower right panel. The upper left panel clearly shows that the measured signal drops with increasing aperture, due to the the underestimated sky value as expected from the normalisation of the data by dividing with the aperture mean. The panel from the sky annulus corrected photometry shows the opposite trend, an increase of about half a milli modulation amplitude when going from an aperture radius of 10 to 25 pixels. This is not surprising, since it corresponds exactly to the increase in the white noise of the FT, as seen in the corresponding panel of Fig. 5.6. The box averaged sky annulus method gives a reasonably sensible value over the entire aperture range.

We can also plot the frequency of the 5585 μHz peak in the same way, as seen in

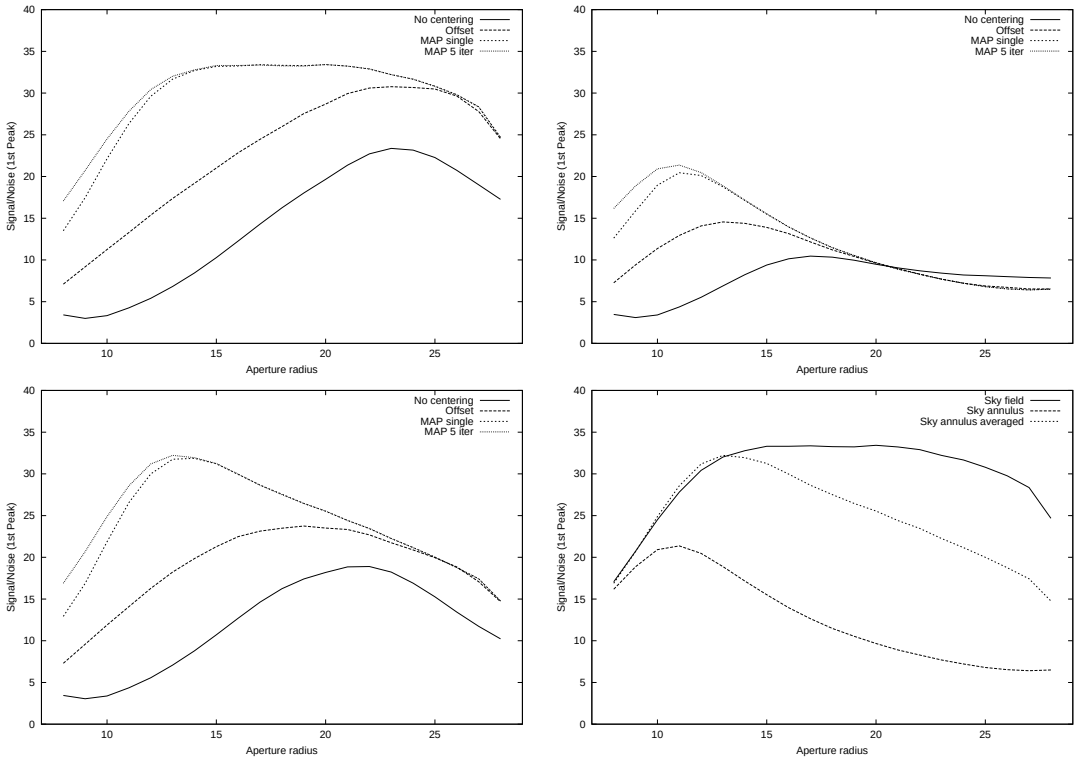


Figure 5.10: PG 1336−018 amplitude of the main peak divided by spectrum noise, as a function of radius. Panel order is still the same as for Fig. 5.6.

Fig. 5.9. Now, all methods agree on approximately the same value, showing that the peak frequency is not sensitive to the choice of processing method. However, the measured value is about 5590, not 5585 as expected. This error corresponds quite exactly to the known timing error on the order of one millisecond per window per integration. We have processed the data assuming a frame interval of 5 seconds, but we know that the true value is somewhat higher. From the PC clock timestamps we have measured the drift over the whole sequence to be about 5 seconds, corresponding to an error of about 1 millisecond per frame. But we know that the PC clock is not an accurate timekeeper, and can drift more than this during a seven hour run. We have no measurement of this drift, but if the frequency analysis is precise, it appears that the true sequence interval is $5590/5585 = 1.0009$ times higher than the five second interval asked for, *i.e.* an error of about 4.5 milliseconds per frame.

5.3.4 Signal to Noise

Returning now to the question of the signal to noise ratio, we can plot the peak amplitude signal shown in Fig. 5.8 versus the measured noise in the high frequency part of the amplitude spectrum from Fig. 5.6. The resulting fractions are shown

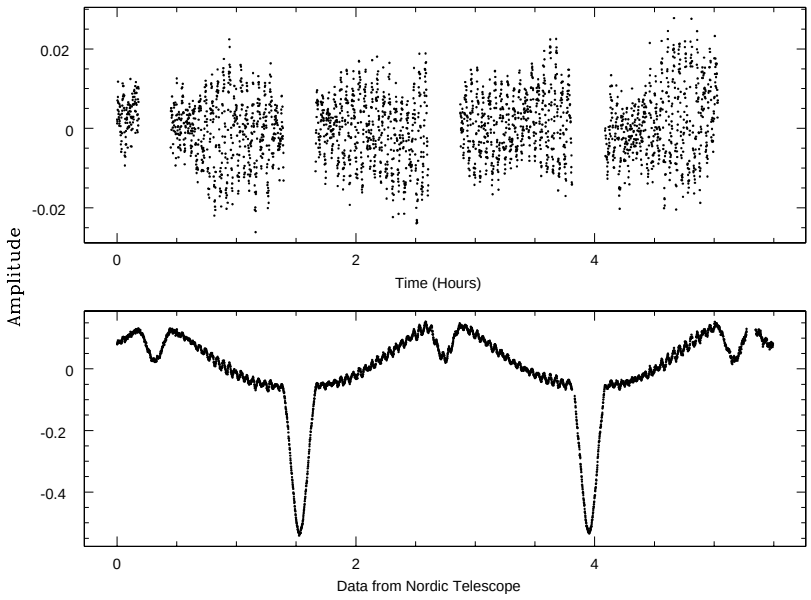


Figure 5.11: PG 1336−018 raw light-curve (bottom) and after removal of eclipses.

in Fig. 5.10. Now the S/N ratio behaves as expected, reaching a clear maximum as the aperture incorporates all the light from the star, and dropping off as more and more sky pixels are introduced into the sum and increasing the noise more than the signal. The sky field method still does not behave well due to the error in the sky level estimation. Even though we have a correct estimate of the signal from the peak amplitude, the noise is still underestimated, since the curve in Fig. 5.6 does not increase above a certain radius as it certainly should.

5.4 XCOV 17 Campaign Results

The PG 1336−018 data have been contributed to the XCOV 17 campaign principal investigator, M. Reed, who has combined them with the earlier campaign data and found them to be of good quality and merging well with the rest of the data set [Ree00]. The plots in Fig. 5.11 were prepared by Reed from our data, and shows how using a special program to remove the complex eclipse features from the data results in a clean light curve. Such analysis is beyond the scope of this work, and in the discussion above we have been content with comparing the straightforward reduction methods implemented in the `rtp` and `rtcorr` programs. Since the eclipses are periodic with a period of about 2.4 hours, they affect mostly the very low frequency part of the spectrum, which has been excluded in the above analysis.

In a multi-site campaign like the WET XCOV 17, much more careful analysis is required to get as much detail as possible from the frequency spectrum. In particular, it is vital to align the sequence as precisely as possible with data from other

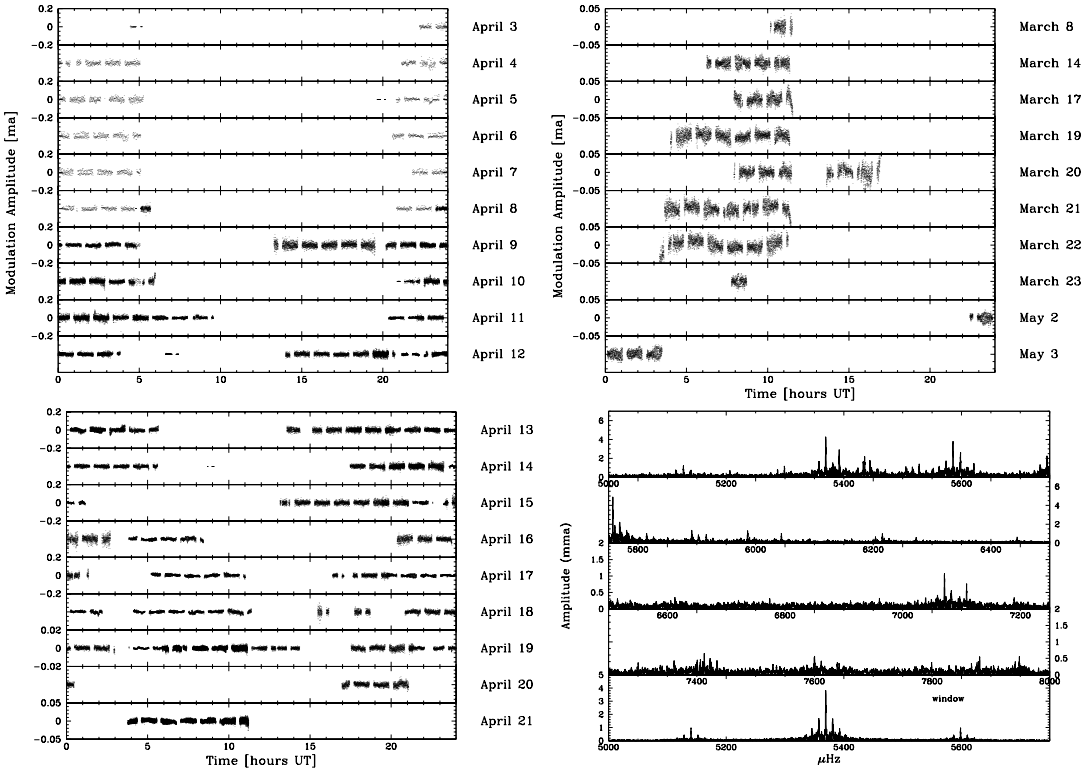


Figure 5.12: XCOV 17 data and section of amplitude spectrum with data from the NOT included (from Reed *et al.* [REE00]).

observatories, something which requires exact timing information. Fig. 5.12 shows all the data from the XCOV 17 campaign on PG 1336−018 (the two left panels), and the pre and post campaign data (upper right panel). The lower right panel shows part of the resulting amplitude spectrum for the combined data set, with the spectral window in the bottom row. Ten observatories around the world contributed during the campaign, our results not included. During the 14 days the campaign lasted, 172 hours of coverage were obtained, plus additional data before and after the campaign (including ours), adding up to a total of 242 hours of observations.

In the discussion of the particular observations obtained during the campaign, Reed *et al.* [Ree00] writes:

> The data from the Nordic Telescope was provided by Roy Østensen in a format adaptable to QED so further reductions proceeded along the standard WET pipeline outlined by Nather *et al.* (1990). The data was received as raw counts from aperture photometry, with several choices for sky measurements available; including an annulus around the target star itself. The integrations were queued up, so they proceeded in equal time intervals, from start to finish. The cycle time (including integration) was 5.001 seconds but the time information stored in the image header was

> based on integration start time. In this case, it was a simple matter to convert the integration start time to the integration mid-point to fit the standard format.

As pointed out here, the standard format requires the time stamps to be integration mid-points, not start of exposure, as have been maintained in all our data files. This could be easily fixed, but we have much more serious problems in the fact that we do not have exact timing information about the start time available, since the time stamps provided by the PC clock is not at all reliable when accuracy of seconds is required.

5.5 In Perspective

Inspection of the individual CCD frames from an observation run where some particular problem was encountered, may not only reveal clearly the cause of the problem, but may inspire some new reduction procedure that can handle it. With the windowed photometry mode each data file is reduced in size from 2 Mb to 40–80kb, making it possible to fit the data from a whole night of 5 second exposures with eight 64x64 windows on a single CD-ROM. Thus, continuous CCD photometry in windowed mode produces no more data per night than normal CCD full-frame imaging with exposure times on the order of 10 minutes, even when all the individual FITS window frame images are stored.

We have seen that our system performed adequately for providing data for the XCOV 17 campaign of the WET. But in spite of this, I consider the timing problem to be our most serious problem with our CCD photometry system. The lack of exact timing information, and the absence of a simple way to set an observation run to proceed from a predefined time (*e.g.* exactly on a full minute), may not be vital for short single observation runs. But for observations that continue over several nights, or in multi-site campaigns, the demand on accurate timing can become more than we can provide with the current system. It will be vital for the future use of the system to provide a good solution to this problem. The best way would be to include a GPS interface to the CCD controller, and have the system automatically read the GPS clock and provide precise timestamps to the user system. This would require some more hardware development (a minor upgrade to the Optio board) and more controller software extensions.

Chapter 6

A Real Observation Run

After the sucessful test of the Tromsø CCD Photometer at NOT in May 1999, using the filter wheel and shutter for the HiRAC camera, development of our own filter and shutter unit continued over the summer. The software for real time photometry were rapidly improved, and by September our extensive testing showed that the system was working as intended, and that we were ready to take it on a real observation run. In October, 1999 we had observation time at the NOT for a search programme to investigate subdwarf stars for variability, and had a good opportunity to test the the multi-window CCD photometry system. Since we were not completely happy with the prescision of our shutter, we decided to leave our photometer at home and use the NOT HiRAC system instead. We already knew that we could interface our program to the filter and shutter unit on HiRAC, and the setup for the Loral 2k chip had already been prepared, so we expected to get the system up and running in multi-windowing mode in a matter of a day or two.

This chapter describes the exellent results obtained on this observation run, starting with a short description of the search programme, then presenting our results on four clearly pulsating sdB stars; one found on an earlier search programme run (PG 1618+563B), and three more discovered during this run.

6.1 The search programme

A programme to search for variable sdB stars with the Nordic Optical Telescope was first suggested in 1998 by Roberto Silvotti of the Osservatorio Astronomico di Capodimonte in Napoli, Italy. He was collaborating with a German group that had performed follow-up spectroscopy on a long list of blue objects based on the Palomar–Green catalouge [Gre86]), and the Hamburg–Schmidt and Hamburg–ESO surveys [Heb99], originally used for identifying quasars. A list of more than 80 spectroscopically identified, previously unknown, sdB and sdOB stars with estimates

of $T_{\rm eff}$ and log g, had been compiled, and provided an excellent opportunity for us to discover new variables. The Tromsø group went into this collaboration, applying for observation time at the NOT in 1999. Since our targets are relatively bright ($13 < m_B < 17$), we could accept time allocation during *grey time* – the weeks when the moon is between a quarter and three quarters full. Time was granted first in July 1999, and later on three more occations.

Spectroscopy of the stars on our target list were done with the 3.5 m telescope of the Calar Alto Observatory in 1988, getting medium resolution (2.5 Å) spectra covering the range from 4100 to 5000 Å. Technical details on the observations and data reductions were presented by Moehler *et al.* [Moe90], together with spectra of 92 stars from the Palomar–Green (PG) Catalog, among these PG 1618+563B. For spectroscopic details about the sdBVs found in this run, see our paper [OEs00b].

The first observations of the search programme were completed in July 1999, using the Tromsø-Texas photometer. Over four nights, 13 objects from the compiled target list were observed, and one pulsator, PG 1618+563B, was detected [Sil00a, Sil00b].

Based on estimates from the South-African group, only about 10% of sdBs in the appropriate temperature/gravity range show pulsations [ODo99], so we expected to have between 8 and 10 pulsators on our list. With the increased effectivity the CCD observations allow us, we expect to finish our list of candidates in only three more runs. The results presented in this chapter are based on the second run of the search programme completed in October, 1999. For this run we used the HiRAC CCD system, with the software described in the previous chapter for the observations. In 2000, we have obtained time for the two final runs of the search programme, but the results have not yet been completely analysed. The real time processing revealed two more pulators during the July 2000 run, and the final run in October, 2000 will be held after the completion of this work.

6.2 Setting up the system

In July, 1999, having proved that our system worked as intended, we decided to attempt use the HiRAC with its normal LN_2 cooled Loral, Lesser thinned, 2048×2048 science detector. The only hardware change required to run multi-windowing photometry was to substitute the sequencer board with an identical spare board that was equipped with an EP-ROM containing the download module, instead of the standard Loral 2k code. Again, we used the `download` program to transfer the camera controller software described in chapter 3, but now compiled for the Loral 2k chip and with configurations for the previous version of the clock driver board, bias generator board and CDS board. The software worked as planned, although an offset in the bias level had been introduced by changing the order of initialisation of the clock drivers. This problem was easily fixed by restoring the original initialisation order. Another problem discovered was that the synthetic overscan mode (produced

by sampling pixels without shifting the serial register) that can be invoked by sending an `@OVER` command to the controller (see table 3.3) produced a higher bias level than normal overscan (produced by sampling pixels after the last real pixel have been shifted off the frame). This problem is probably also caused by improper initialisation of the drivers, but was not investigated further, since synthetic overscans were not required for our purposes.

The HiRAC's CCD chip has several problems. Among these is a large number of surface defects that litter the chip surface. Most of these are in the form of "droplets", some 4 by 4 pixels large, where the sensitivity drops by as much as 50% in the center. There are also three bad columns, where serious charge trapping occur, extending from faulty pixels and up to the top edge of the frame. We made a sincere effort during the setup of each observation run to avoid these bad regions, when the windows for target and reference stars were selected, even if this meant that we had to offset and re-expose the field several times. Unfortunately, in some cases we did not manage to spot the problems in the field frame and a few of our observations suffer from CCD pixel related problems. The surface defect problem is a considerable drawback of the HiRAC system, since it slows down the setup process for starting an observation series on a new target, and introduce the potential risk of seriously degrading the observations if care is not taken during all setups. The TK1024 chip does not suffer from such problems.

The sky area available for locating a reference star is limited to the area of the HiRAC Loral 2k chip, and corresponds to $\sim 3.7\times3.7$ arcminutes^2. In the 3^{rd} run of the search programme, in July, 2000, we decided to use the Andalucia Faint Object Spectrograph and Camera (ALFOSC) for the observations, since this instrument contains a focal reducer that gives a field of $\sim 6.5\times6.5$ arcminutes^2. The camera is otherwise similar to the HiRAC camera, and the Loral 2k chip shows the same surface defects. The defects are due to the fact that the Loral chips are not delivered in thinned versions from the factory, but are thinned by Mike Lesser of Steward Observatory. His ultra-thinning process produces excellent sensitivity in the ultraviolet, but the uniformity of the surface suffers.

Table 6.1: Targets observed during sdBV search programme's 2^{nd} run.

Target name	R.A.	δ	Epoch	m_B^a	m_B^b
HE 0021-2326	00 21 28.5	-23 26 31	B1950		16.0
HE 0123-2808	01 25 33.4	-27 53 04	J2000	16.0	15.8
HE 0324-2529	03 24 06.0	-25 29 03	B1950		14.2
HE 0405-1719	04 05 12.0	-17 19 12	B1950		13.7
HE 0429-2448	04 31 28.3	-24 41 57	J2000	15.1	14.9
HE 2205-1952	22 08 41.3	-19 37 39	J2000	14.5	14.2
HS 0023+3049	00 23 24.5	+30 49 44	B1950	14.6	14.7
HS 0035+3034	00 35 58.6	+30 34 14	B1950	15.8	15.9
HS 0048+0026	00 48 33.1	+00 26 17	B1950	15.6	
HS 0055+0138	00 55 50.3	+01 38 22	B1950	15.1	14.8
HS 0213+2329	02 13 05.5	+23 29 19	B1950		13.4
HS 0232+3155	02 32 21.0	+31 55 53	B1950		15.1
HS 0546+8009	05 46 27.8	+80 09 55	B1950		14.0
HS 0600+6602	06 00 21.7	+66 02 31	B1950		16.4
HS 0815+4243	08 15 53.2	+42 43 04	B1950		16.1
HS 2149+0847	21 49 26.1	+08 47 14	B1950	16.5	16.4
HS 2151+0214	21 51 24.7	+02 14 15	B1950	16.1	16.1
HS 2156+2215	21 56 21.1	+22 15 30	B1950	15.1	15.4
HS 2201+2610	22 01 54.4	+26 10 33	B1950	13.6	14.3
HS 2206+2847	22 06 05.0	+28 47 57	B1950	14.2	
HS 2208+2718	22 08 19.2	+27 18 03	B1950	14.1	14.9
HS 2209+2840	22 09 47.0	+28 40 25	B1950	14.7	15.6
HS 2225+2220	22 25 09.9	+22 20 51	B1950	15.5	15.9
HS 2229+0910	22 29 55.3	+09 10 24	B1950		14.8
HS 2242+3206	22 42 43.5	+32 06 04	B1950	14.8	13.8
HS 2333+3927	23 35 43	+39 44 28	J2000	14.5	14.3
PB 6740	02 12 25.0	+01 55 09	J2000	13.7	13.9
PG 1618+563	16 19 13.6	+56 08 23	J2000	11.3	
PG 2233+1418	22 35 31.3	+14 33 55	J2000	13.4	13.8
PG 2240+105	22 43 29.0	+10 46 42	J2000	14.8	14.7

6.3 Targets Observed

The list of 31 sdBs actually observed during the 2^{nd} run of the search programme is presented in Table 6.1. The HE designations are from the Hamburg–ESO Survey, *i.e.* the southern hemisphere equivalent of the HS Survey [Wis91]. PB 6740 has its designation Palomar–Berger survey [Ber80], but appears also in the PG catalog.

We observed the stars in windowed mode using two or more reference stars for

constructing the relative light curves. The observations were in general made with the HiRAC B-band filter, and with a 20 s cycle time. The actual exposure times varied as we started out with five 64×64 pixel windows (target, two reference stars and two sky fields) and an exposure time of 14.7 s, and later changed this to six 42×42 pixel windows (one additional reference star) and an exposure time of 15.8 s.

The positions and magnitudes (m_B^a) in Table 6.1 have been estimated from the photographic Schmidt plates of the HS survey. The m_B^b column give our estimates of the magnitudes from the CCD photometry. No attempt at absolute calibration has been made, since these magnitudes are only used as a guide to evaluate the noise in the final light curve. The zero point of the magnitude scale has been arbitrarily scaled to get a best fit to the HS magnitudes. As we can see the fits mostly correspond within the errors of ±0.25 mag given for the HS values, but for a few of the targets the error is almost a full magnitude. When estimating our magnitudes we have attempted to fit a line trough the extinction corrected light curves, avoiding obvious cirrus contamination. During some observations we see continuous cirrus activity, and therefore these estimates may be a bit on the low side.

6.3.1 Through the Grinder

The results presented in the following have been through many levels of processing. Although the `rtp` program clearly revealed the pulsations in the three identified variables as the data were coming in, all photometry have been recalculated to obtain the best results possible. A careful examination has been performed on each of the targets were we saw nothing during the observations, to see if something could be hiding in the noise. Some suspicious targets have been identified and flagged for rechecking at the next opportunity, but no more pulsators were found.

To process the pile of images from our 31 targets, three useful programs in the Perl scripting language have been written. They perform reprocessing, corrections and Fourier analysis, and noise comparison of the results. The first script loops through all the directories set up for `rtp` and invokes `rtp` 12 times for each target, using aperture radii of 10 to 21 pixels, and stores the resulting photometry from each calculation as independent files. These synthetic apertures correspond to aperture diameters of 2.2 to 4.6 arcseconds. The second script loops over all the photometry files, calling `rtcorr` to perform extinction correction, differential photometry, normalisation and windowing, and calling `rtft` to compute amplitude spectra of the windowed differential photometry. The final script runs through all these output files and computes noise parameters. A summary file is generated for each data set that tabulates average signal values, signal to noise ratios and white noise terms (amplitude spectrum average value), as a function of aperture radius. Based on these tables, it is easy to determine the optimal aperture radius (and the best reference star) to use for each individual target.

This automatic approach has been very helpful in making it easy to reprocess all the data again and again, as the software improved and bugs were chased out. But some of the targets reveal odd behavior, *e.g.* higher noise than expected from the signal and run lengths. In these cases the individual frames have been carefully examined with `rtcnv` to reveal the origin of the problem. Discussions are given for each of the individual targets below. First we will present the results and discussions on the variable sdBs, before we go on to a detailed examination on the 28 remaining sdBs.

6.4 Pulsators Found

Investigations of our observations turned up three new sdB variables. These are HS 0815+4243, HS 2149+0847 and HS 2201+2610 (hereafter HS 0815, HS 2149 and HS 2201 for short). Thus, 3 out of 28 new sdB stars investigated were found to be variable. Additionally, PG 1618+563B (hereafter PG 1618) was observed again for confirmation and comparison with the PMT data, and HS 2206+2847 was also re-observed since the PMT observations of the 1^{st} search programme run could not rule out the possibility of pulsations. We will start by presenting the observations on PG 1618.

6.4.1 PG 1618+563B

The PG 1618 system was setting very early, so we tried to catch it as soon as possible after dark on the second night of our observation run. We intended to observe the target in the V-band, but due to a mistake the R filter were used. The weather conditions were poor, so we decided to go on as long as possible without much hope of getting useful data. The target was observed with a cycle period of 20 seconds, and we accumulated 1 hour and 25 minutes worth of data, before the target disappeared totally in the clouds. The result was clearly the poorest set of data of the whole run, and show an example where the differential aperture photometry method really has problems, as we can see from Fig. 6.1. The reason for the difficulties was the fact that the only reference star available in the field, as limited by the CCD chip's 3.75 arcminute field of view, was the close companion PG 1618+563A. The distance between the two stars is only 3.7 arcseconds, or about 4 times the full-width half-maximum of the stellar images. With such a close proximity it is inevitable that a small fraction of the light of the reference star contaminates the target, and *vice versa*. Under cirrus conditions, or whenever the seeing changes during the observations, the degree of contamination varies with the seeing. In this case, most likely, PSF photometry would have solved the problem. I have not attempted to implement a full PSF algorithm into the `rtp` program, but we note that there are cases where it would be useful to be able to reprocess the data using PSF profile fitting.

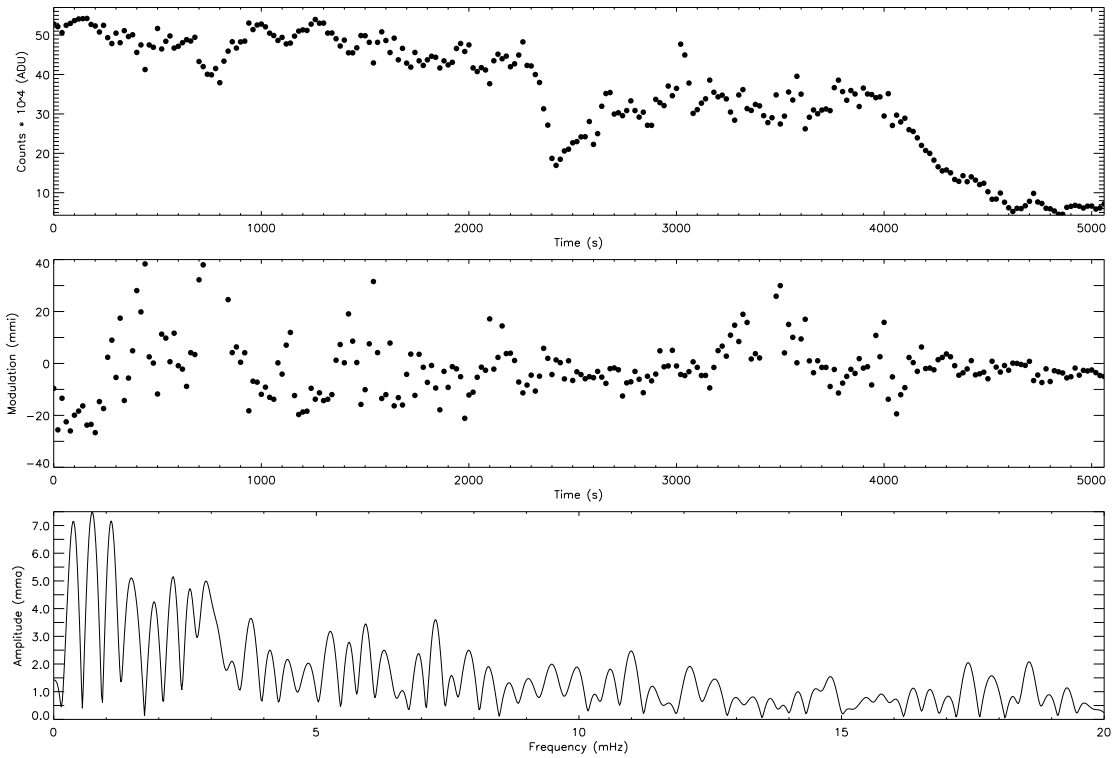


Figure 6.1: R band light curve (extinction corrected and sky subtracted), differential photometry and amplitude spectrum for PG 1618+563B, observed through cirrus clouds and at low altitude.

Observations on the next night were collected in the U and B bands, with much higher success than the night before. The atmospheric conditions were a lot more stable, allowing us to use the companion star as a reference star without much interference. Details about the PG 1618 observations are given in the upper part of Table 6.2, below. Observations are indexed by band (R, U, B), and in addition to giving the observation date, start and stop times, the table lists the number of frames N, the exposure time T_e, the readout time, T_{ro} and sequence time interval, T_s.

The companion star has spectral class F3, so it is considerably redder than the target. In the B band they have comparable magnitude, but in the U band the target is 25% brighter than the companion. Also, the target gets 3 times more counts in the blue than in the ultraviolet, so the noise is of course significantly higher in the U than in the B. Still, the U band light curve and amplitude spectrum shown in Fig. 6.2, clearly reveals pulsations at the expected frequencies. The pulsations are, as were also seen in the PMT data, weaker in the B band observations shown in Fig. 6.3, than in the U band.

The PMT observations from the first run revealed peaks at about 6948 and 7180 μHz. The amplitudes were found to be about 4.6 and 1.5 in the U band, and slightly

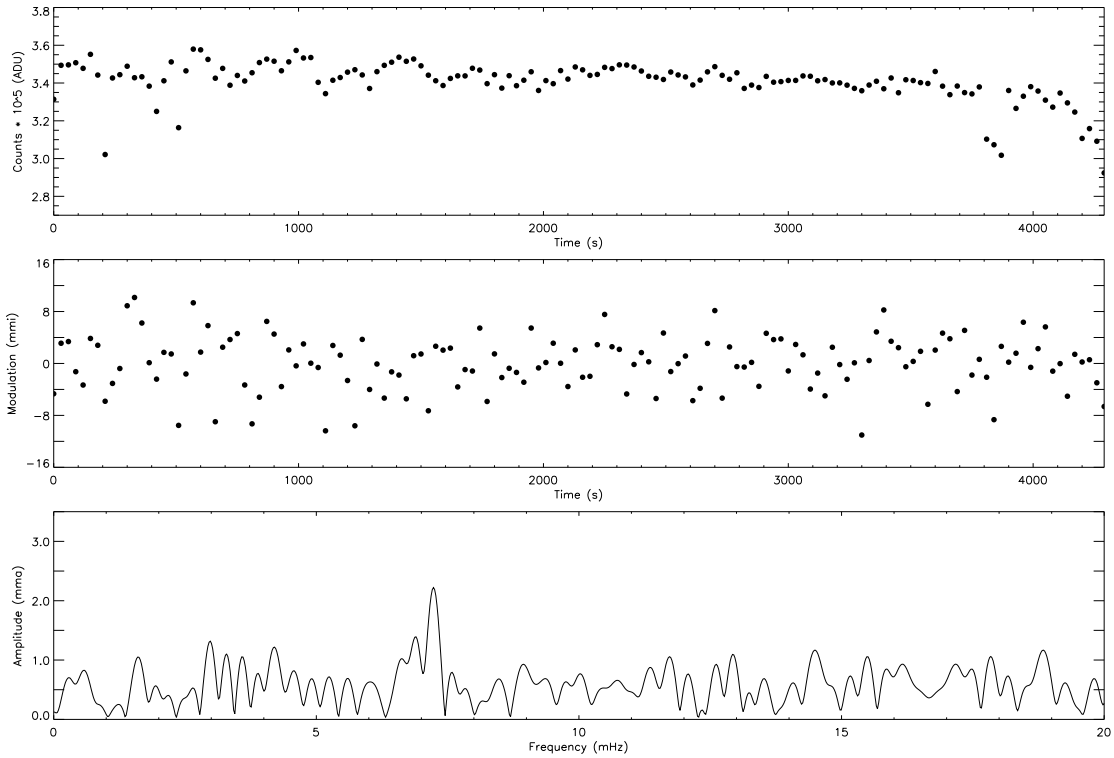


Figure 6.2: U band light curve, differential photometry and amplitude spectrum for PG 1618+563B.

less, about 4.3 and 1.2 in the B band. Observations in the V band on two consecutive nights revealed that the amplitudes, at least for the weaker period, were changing significantly from night to night [Sil00b]. Our CCD observations reveal that the amplitude of the first period has dropped significantly, while the second peak has come up as the strongest. The U observations show an amplitude of 1.4 mma for the first peak and 2.2 mma for the second peak. For the B band, only the second peak is significant above the noise, at an amplitude of only 1.7 mma. The results are summarised in Table 6.2, below. The noise estimates σ_{0-6} and σ_{8-20} are the means of the amplitude spectra in the ranges 0 – 6 000 μHz and 8 000 – 20 000 μHz, respectively. For the R band light curve there is a significant $1/f$ noise component, so the actual noise around 7 000 μHz is close to 2 mma, which means that the peak at 7 270 μHz, which corresponds to the main peak revealed in the U and B data, is not significant.

For a thorough discussion on the results for PG 1618, see our paper [Sil00b].

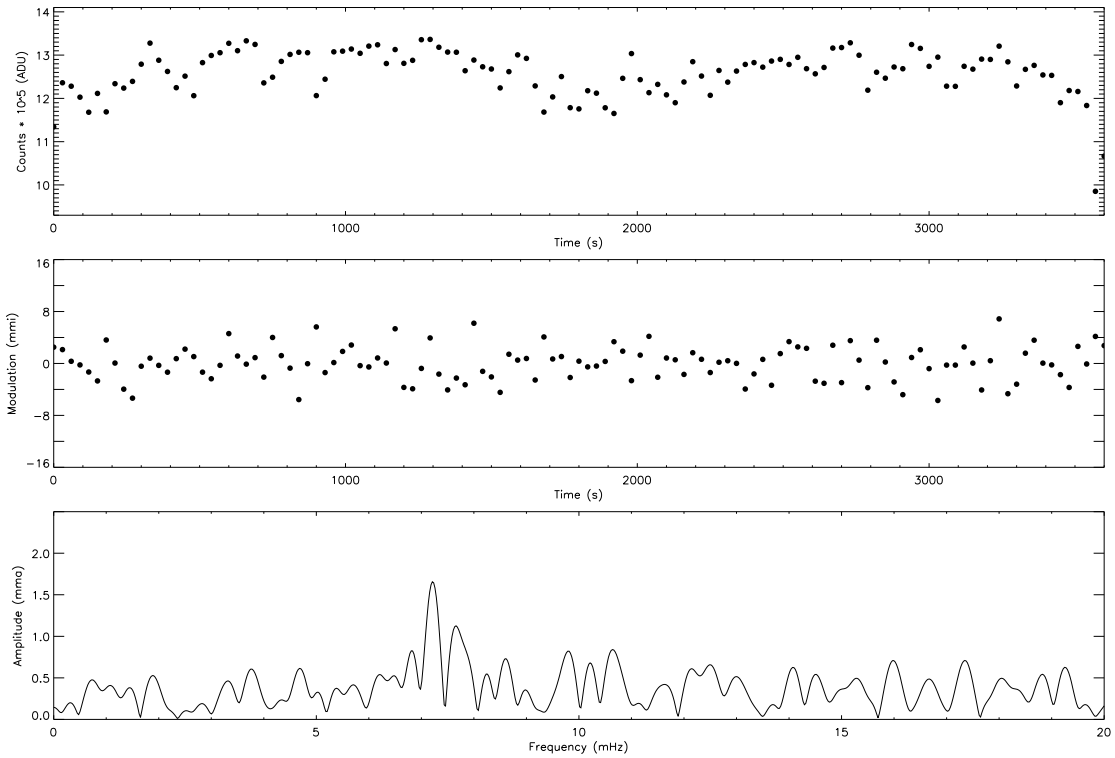


Figure 6.3: B band light curve, differential photometry and amplitude spectrum for PG 1618+563B.

Table 6.2: Observations and reductions for PG 1618+563B.

Run	Date	start	end	N	T_e [s]	T_{ro} [s]	T_s [s]
R	1999-10-15	20:19:36	21:44:12	254	15.20	4.42	20.0
U	1999-10-16	19:24:29	20:36:28	144	26.20	3.51	30.0
B	1999-10-16	20:38:29	21:38:58	121	26.20	3.51	30.0

Run	f_0 [μHz]	P_0 [s]	A_0 [mma]	f_1 [μHz]	P_1 [s]	A_1 [mma]	σ_{1-6} [mma]	σ_{8-20} [mma]
R	7269	137.58	3.60				2.90	0.85
U	7231	138.30	2.22	6889	145.15	1.39	0.53	0.53
B	7215	138.60	1.65				0.28	0.33

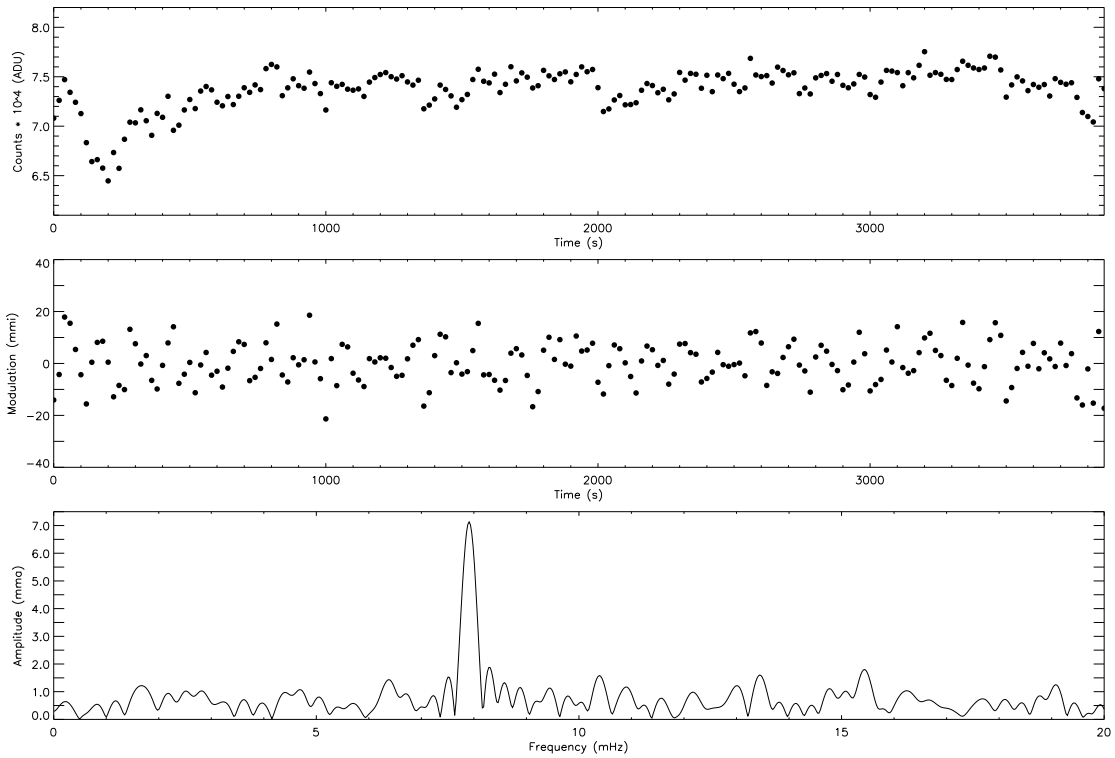


Figure 6.4: Sky subtracted, extinction corrected light curve (upper panel), normalised differential photometry (middle panel), and amplitude spectrum (lower panel) for the discovery run on HS 0815+4243.

6.4.2 HS 0815+4243

On the second night of our run, just before dawn on October 16^{th}, we found the first pulsating sdB star in our sample. HS 0815 was observed again on the following two nights to confirm the detection of pulsations. The details of the observation start and stop times, number of frames, exposure times, readout times and sequence times are listed in table 6.3.

A search in the SIMBAD Database[1] reveals that HS 0815 has independently been classified as a faint blue star by the First Byurakan spectral sky survey (FBS B 251 = FBS 0815+427 [Abr90]), and as a NHB (normal or horizontal branch star) by the Kiso survey (KUV 08159+4243 [Weg86]).

For the first night, an optimal aperture radius of 14 pixels (∼3.1" diameter) was chosen, based on the signal to noise ratios and amplitude spectrum white noise. We use the differential photometry based on the sum of the two available reference stars, which gives us a reference level that is about 1.4 mag brighter than the target,

[1]SIMBAD Astronomical Database at Centre de Données astronomiques de Strasbourg (CDS); http://simbad.u-strasbg.fr/

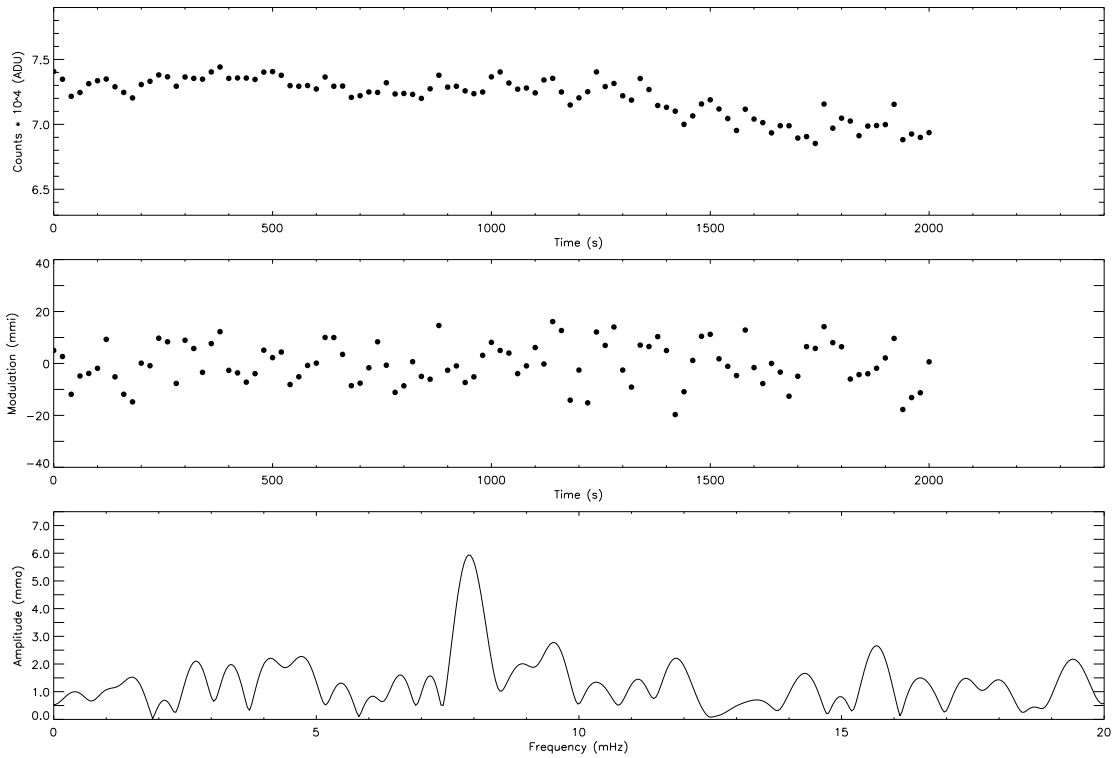


Figure 6.5: Sky and extinction corrected light curve (upper), normalised differential photometry (middle), and amplitude spectrum (lower) for the 2^{nd} run on HS 0815.

and produces a light curve and amplitude spectrum as shown in Fig. 6.4. For the second night, the run was started late in the night, and became too short to get a fair resolution in the temporal spectrum. The best results can be found when using an aperture radius of 13 pixels ($d = 2.86$"), and the first reference star only, as in Fig. 6.5. For the third night, we get optimal results with an aperture radius of 12 pixels ($d = 2.64$"), and again using the first reference star, as shown in Fig. 6.6.

All the amplitude spectra show one clear period close to 7 900 μHz. Table 6.3 shows the results of the Fourier analysis for these observations. The columns f_0, P_0 and A_0 gives the frequency, period and amplitude of the main period. The columns with σ_{0-7} and σ_{9-20} gives the average white noise level of the spectrum in the low frequency range between 0 and 7 000 μHz and the high frequency range between 9 000 and 20 000 μHz respectively. As we see, the noise level is below to 0.7 mma for both the first and third run, and below 1.2 mma for the second run.

The light curves show signs of modulations, which implies that at least two closely separated periods are present. All the three amplitude spectra show a dominant peak at 126 s with an amplitude of about 7 mma. The peak at 120 s is barely significant in the first run at 1.9 mma, but not seen in the third. In the third run one can see a second barely significant peak at 131 s with an amplitude of 1.9 mma

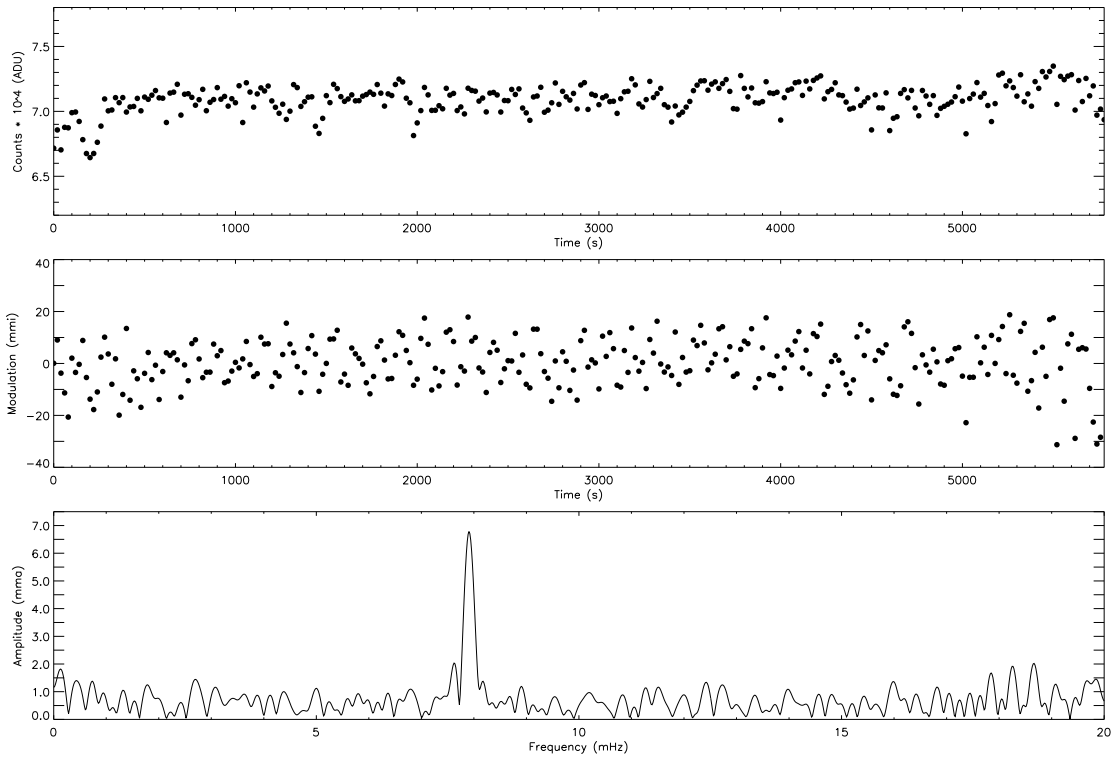


Figure 6.6: Sky and extinction corrected light curve (upper), normalised differential photometry (middle), and amplitude spectrum (lower) for the 3^{rd} run on HS 0815.

(Fig. 6.6), which is also present, but not significant, in the first run (Fig. 6.4). A Fourier transform of the window function was computed by Silvotti for our paper [OEs00b], and indicates that these peaks most likely are artifacts of the spectral window. A longer run would be required to resolve the temporal spectrum of this star.

Table 6.3: Observations and reductions for HS 0815+4243.

Run	Date	start	end	N	T_e [s]	T_{ro} [s]	T_s [s]
1	1999-10-16	05:23:40	06:28:20	194	14.70	4.93	20.0
2	1999-10-17	06:00:20	06:34:00	101	14.70	4.93	20.0
3	1999-10-18	05:04:00	06:31:20	262	14.70	4.93	20.0

Run	f_0 [μHz]	P_0 [s]	A_0 [mma]	σ_{0-7} [mma]	σ_{9-20} [mma]
1	7909	126.44	7.13	0.62	0.65
2	7910	126.43	5.94	1.16	1.13
3	7908	126.45	6.78	0.69	0.65

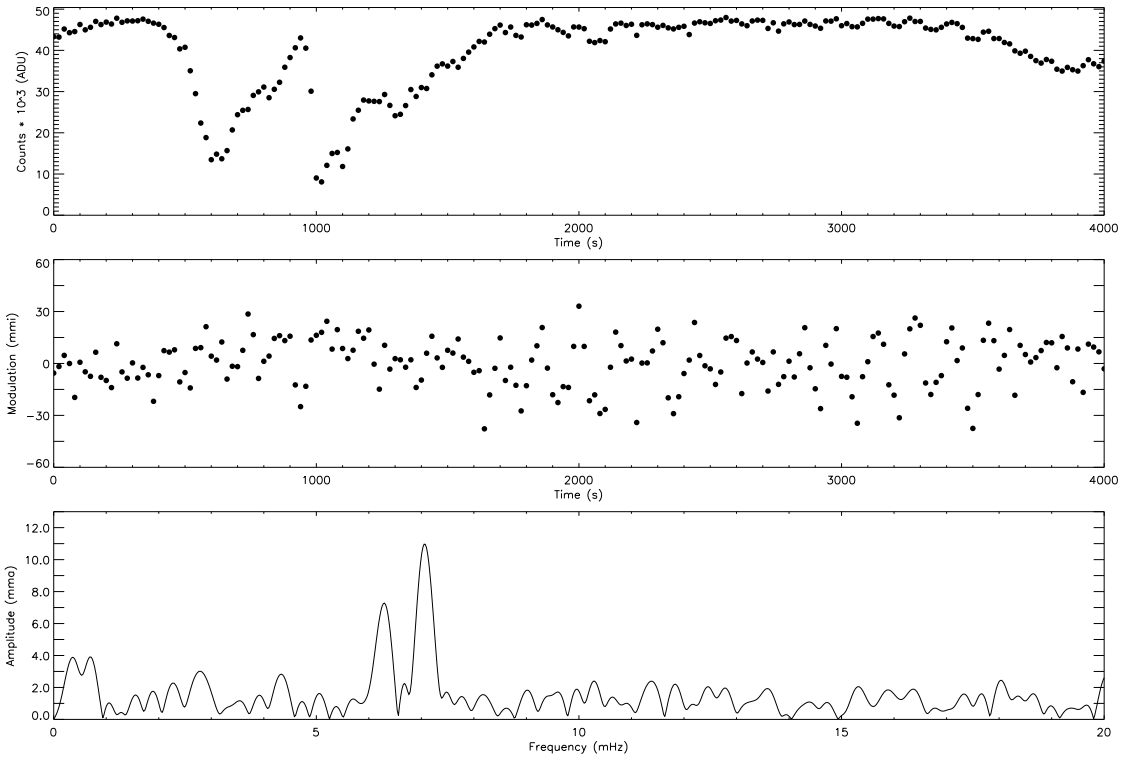


Figure 6.7: Sky subtracted, extinction corrected light curve (upper panel), normalised differential photometry (middle panel), and amplitude spectrum (lower panel) for the discovery run on HS 2149+0847.

6.4.3 HS 2149+0847

This object was identified as a variable on October, 17^{th}, and clearly shows two periods at 141 and 159 s, with amplitudes of about 7 and 11 mma respectively. There is an indication of a third frequency between these at about 150 s, showing an amplitude of 2.2 and 3.3 mma in the first and second night respectively, but this is doubtful considering the short time coverage. In the light curve we can clearly see the beat period of about 1200 s (see middle panel of Fig. 6.7 and Fig. 6.8).

Run 1 was done using 5 windows (target, two reference stars and two sky fields) with 64x64 pixels, while run 2 was made with 6 windows (one more refreference star) and the window size reduced to 42x42. With this reduction in window size, even when adding one more window, we save about a second on the readout period and thereby improving the duty cycle from 73.5% to 79 %. See the upper part of Table 6.4 for details on the observation date, run length, and sequence timing information.

For this target the noise analysis show that the optimal results are obtained with aperture radii of 11 and 12 pixels for the first and second night respectively. For the first night we observed with two reference stars; the first being about one magnitude

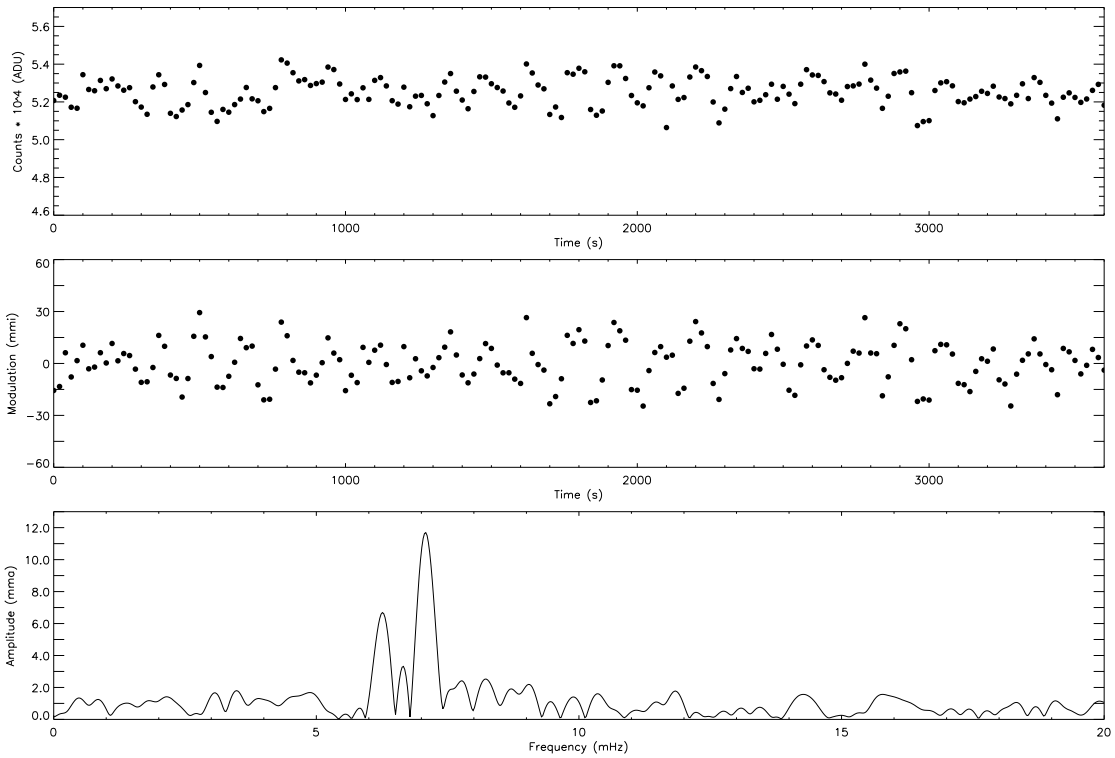


Figure 6.8: Sky subtracted, extinction corrected light curve (upper panel), normalised differential photometry (middle panel), and amplitude spectrum (lower panel) for the 2^{nd} run on HS 2149+0847.

brighter than the target, the second only 0.1 magnitude brighter. The second night we observed with three reference stars. Noise analysis show that the best results are obtained using only the bright reference star for the first night and the sum of all three reference stars on the second. The periods, amplitudes and white noise levels are given in the lower part of Table 6.4. As we see, the noise level is about 1.2 mma in the first night and 0.9 mma in the second.

Table 6.4: Observations and reductions for HS 2149+0847

Run	Date	start	end	N	T_e [s]	T_{ro} [s]	T_s [s]
1	1999-10-17	23:25:00	00:32:00	201	14.70	4.93	20.0
2	1999-10-18	22:13:00	22:57:00	101	15.80	3.88	20.0

Run	f_0 [μHz]	P_0 [s]	A_0 [mma]	f_1 [μHz]	P_1 [s]	A_1 [mma]	σ_{1-6} [mma]	σ_{8-20} [mma]
1	7062	141.60	10.98	6292	158.94	7.28	1.20	1.15
2	7077	141.29	11.70	6260	159.73	6.69	0.94	0.86

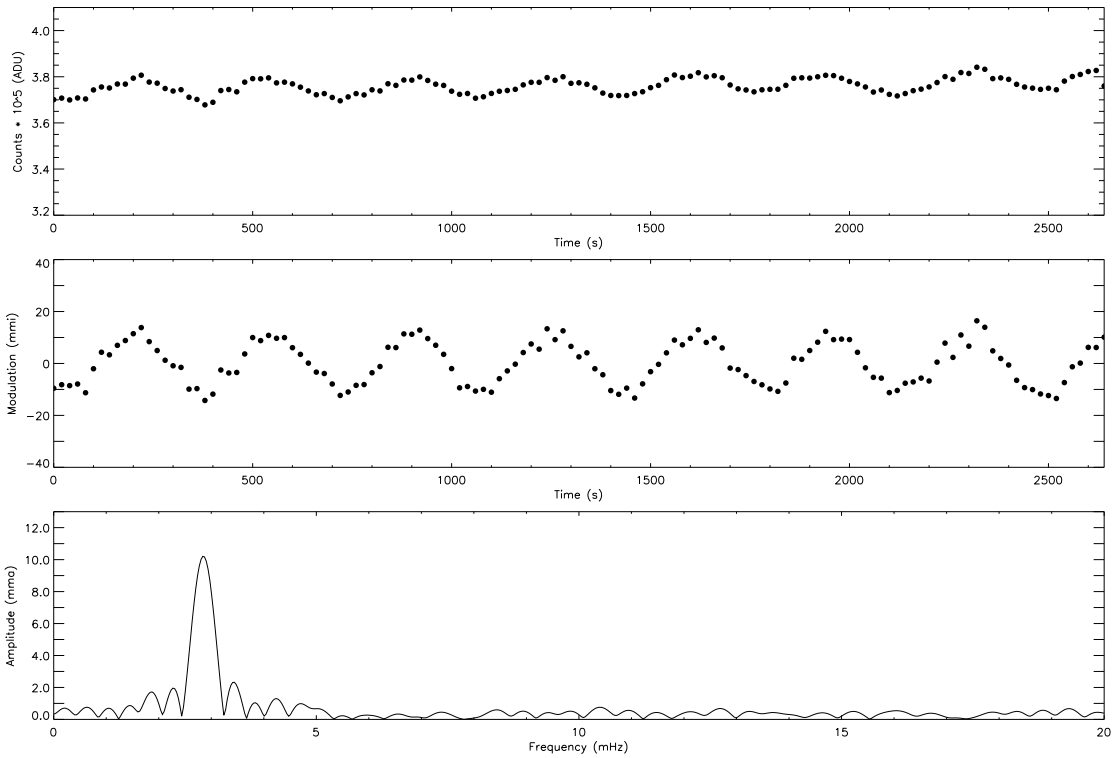


Figure 6.9: Sky subtracted, extinction corrected light curve (upper panel), normalised differential photometry (middle panel), and amplitude spectrum (lower panel) for the 2^{nd} run on HS 2201+2610.

6.4.4 HS 2201+2610

This object was discovered to be variable during the last observing night, and even after only ten minutes of observations it was clear that HS 2201 is a pulsator. The main period is at 351 s, with an amplitude of 10 mma. Since the run was only 40 minutes long, the resolution of 0.4 mHz may hide considerable fine structure, as it can be seen in Fig. 6.9. Noise analysis gave the best results when processing with an aperture radius of 15 pixels, and taking the differential photometry between the target and the sum of all three available reference stars. Two of the reference stars were of the same brightness as the target, the third was half a magnitude brighter. Thus, the total comparision magnitude used here was about 1.5 magnitudes brighter than the target. The noise level in the higher frequencies (above 5 mHz) is below 0.34 mma, and even though there are several periods with amplitudes in the range 1 – 2 mma in our FT, we cannot determine their significance based on 40 minutes of observations only.

The extinction corrected light curve gives $B = 14.3$ for this target, which is not in very good agreement with the magnitude estimated from the original Hamburg Schmidt plates. Since we had little cirrus on this night, as is clear from the extinction

Table 6.5: Observations and reductions for HS 2201+2610

Date	start	end	N	T_e [s]	T_{ro} [s]	T_s [s]
1999-10-18	23:20:58	00:04:58	132	15.80	3.88	20.0

f_0 [μHz]	P_0 [s]	A_0 [mma]	σ_{5-20} [mma]
2849	351.00	10.22	0.33

corrected light curve (top panel of Fig. 6.9), we believe our estimate is the better one.

This object has become the primary target of a multi-site campaign running in September–October, 2000.[2]

6.4.5 Summary on the Pulsators

The three sdB stars discovered to be variable, which are located well inside the region of the ($T_{\rm eff}$, $\log g$) plane where the sdB pulsators are expected, have quite simple temporal spectra, as far as we can judge from our short light curves.

HS 2201 has almost the same $T_{\rm eff}$, $\log g$ and pulsation period as Feige 48, which shows at least 4 periods with variable amplitudes [Koe98c]. For Feige 48 the strongest pulsation at 2.87 mHz can vary between 4.4 and 13 mma from one night to the next. Although HS 2201 appears to have only a single pulsation, the frequency of 2.85 mHz and amplitude of 10 mma matches the main pulsation of Feige 48 perfectly, so the connection is quite clear. Since we investigated HS 2201 only during one night, we had no opportunity to check if its amplitude is variable. The low resolution of the FT does not rule out multiple periods close to the main period.

The other two hot pulsators near $T_{\rm eff} = 35\,000$ K, are placed just on (for HS 2149) or below (for HS 0815) the locus of the fundamental mode, in the Period/$\log g$ graph [Fon98], indicating that the pulsations we observed either are associated with the fundamental mode or strong p-modes.

[2] See http://www.na.astro.it/~silvotti/hs_rxj.html for details.

Table 6.6: Observations and reductions of 26 sdB stars.

Target name	Date [1999]	Time [UT]	N_p	N_r	Noise [mma]	Comment
HE 0021−2326	Oct 16	00:22:49	181	2	0.84	No variability
HE 0123−2808	Oct 17	01:10:24	91	1	0.98	No variability
HE 0324−2529	Oct 17	02:42:27	120	2	0.48	No variability
HE 0405−1719	Oct 16	03:30:35	167	2	0.55	No variability
HE 0429−2448	Oct 19	05:11:06	44	3	1.20	No variability
HE 2205−1952	Oct 15	23:10:53	166	2	0.70	No variability
HS 0023+3049	Oct 18	03:13:54	64	2	0.73	No variability
HS 0035+3034	Oct 19	04:26:30	76	3	1.80	Faint & noisy
HS 0048+0026	Oct 15	01:10:12	137	2	0.60	No variability
HS 0055+0138	Oct 17	00:00:10	180	1	0.41	FF error
HS 0213+2329	Oct 16	01:47:28	149	1	0.28	No variability
HS 0232+3155	Oct 17	03:40:48	120	2	0.50	No variability
HS 0546+8009	Oct 17	04:40:21	81	2	1.10	Bad line
HS 0600+6602	Oct 17	05:20:26	99	2	1.20	Possible pulsation
HS 2151+0214	Oct 16	22:53:57	166	2	0.86	Probably nothing
HS 2156+2215	Oct 14	22:08:43	201	3	0.14	No variability
HS 2206+2847	Oct 14	23:54:16	371	2	0.57	No variability
HS 2208+2718	Oct 19	01:30:23	51	2	0.79	FF error
HS 2209+2840	Oct 18	00:39:47	133	2	1.04	Probably nothing
HS 2225+2220	Oct 18	01:31:49	80	2	1.20	No variability
HS 2229+0910	Oct 19	00:16:21	54	3	0.86	No variability
HS 2242+3206	Oct 18	02:18:25	110	2	1.20	Clouds
HS 2333+3927	Oct 19	02:27:50	289	3	0.57	Slow variability
PB 6740	Oct 17	01:51:55	106	1	0.50	Low freq. pulsation
PG 2233+1418	Oct 19	00:45:48	81	3	0.65	Problem with ref. star
PG 2240+105	Oct 19	02:02:40	54	1	1.70	Odd feature at 15 mHz

6.5 Other targets

Apart from the four variable sdBs presented in the previous section the remaining 27 sdBs from table 6.1 were observed at the times listed in table 6.6, above. The table lists targets with the date and time at the start of observations. N_p is the number of points in the sequence, and N_r is the number of reference stars. The "Noise" column gives the mean of the amplitude spectrum, when computed from the extinction corrected differential photometry. All observations were done using the B-band filter, except HS 0048+0026 which were observed without filter. A sequence time interval of 20 seconds were used, except for HS 2206+2847 which was run with a sequence time interval of just 10 seconds. Table 6.6 also shows the noise level

(mean) of the amplitude spectra, as well as comments about suspicious features.

It is worth to note that even though we did not detect any pulsations in our brief time series observations of these sdBs, this does not exclude that they have some variability. In a recent paper Piccioni *et al.* [Pic00] presents observations that shows that the sdB star PG 0856+121 was pulsating at frequencies of 2.3 mHz and 3.2 mHz and amplitudes of 3.0 and 3.5 mmag respectively on two subsequent nights in March 1998. Later observations in April confirmed the pulsations, but when the star was revisited in December 1998 the pulsations were absent or under the detection treshold. PG 0856+121 was investigated spectroscopically in the sample of Moehler *et al.* [Moe90], and found to have a log g of 5.1 and $T_{\rm eff} = 23\,800$ K, but Saffer *et al.* [Saf94] have found fairly different values; log $g = 5.73$ and $T_{\rm eff} = 26\,400$ K. Taking the last values as the most likely, implies that it lies on the cold edge of the sdB instability strip. The star is plotted in Fig. 5.1, where it is clearly seen to be a bit away from the rest. It is likely that this star is also a member of the EC 14026-type variables, although it is unclear what could cause the pulsations to drop so significantly.

From the example of PG 0856+121 it is clear that sdB stars should be monitored much more carefully, and frequently revisited over long periods, in order to say for sure that they are non-pulsators. Thus, an observation run from which we detect no pulsations within four times the noise level at a certain date, may be as valuable for future classifications as a clear detection of variability. Therefore, observations showing with some reasonable limits that such stars show no pulsations should also be published. We are planning to compile a catalouge for the full set of non pulsators into one article, when the observations from all four search programme runs have been processed.

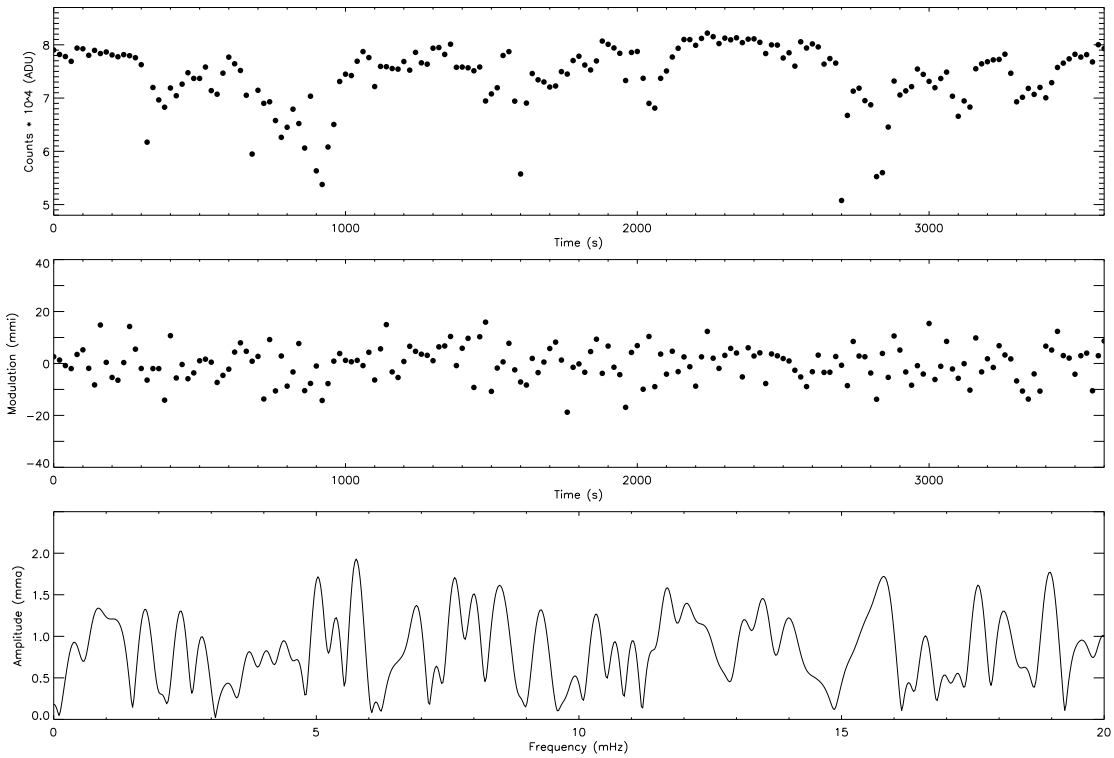


Figure 6.10: Light curve, differential photometry and amplitude spectrum for HE 0021−2326.

6.5.1 HE 0021−2326

This one hour observation run provided a light curve (Fig. 6.10, upper panel) with repeated interruptions by cirrus clouds causing up to 60 % extinction. The differential photometry technique does a very good job in removing the influence of the clouds (middle panel), and the amplitude spectrum of the differential light curve (lower panel) has a mean value of 0.84 milli modulation amplitudes (mma). No significant peaks indicating any periodic pulsations can be seen in the amplitude spectrum.

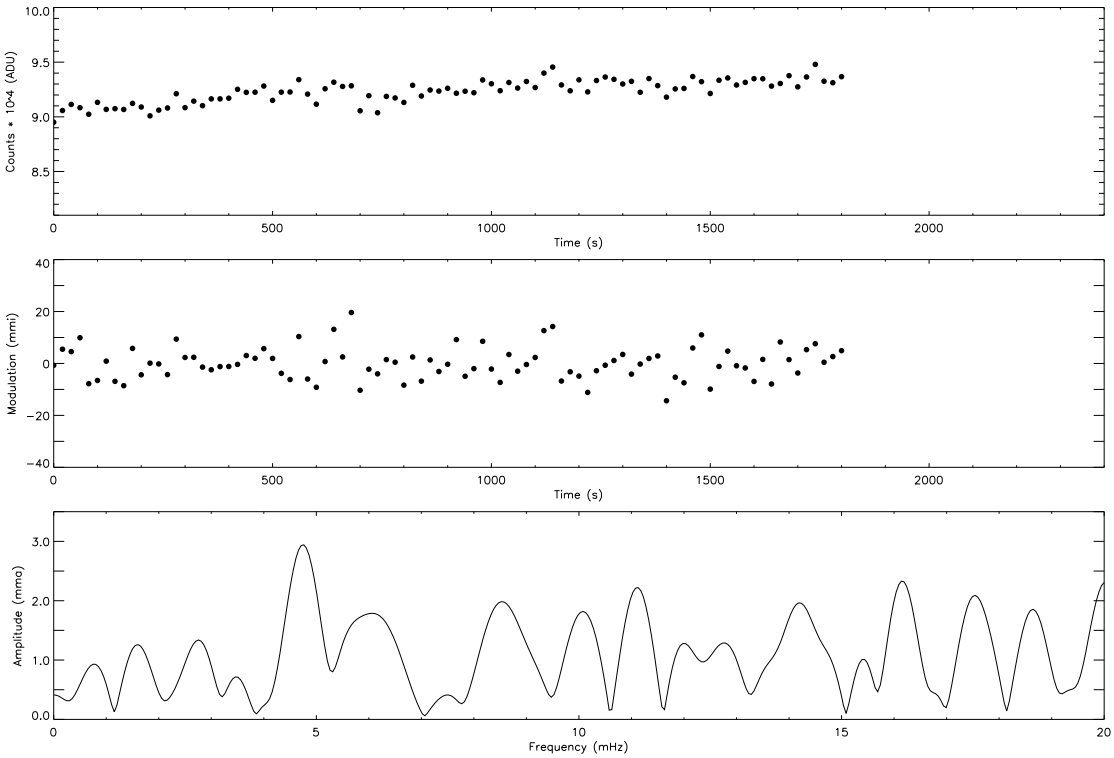


Figure 6.11: Light curve, differential photometry and amplitude spectrum for HE 0123−2808.

6.5.2 HE 0123−2808

This star is also known as GD 1454 [Gic78] and MCT 0123-2808, as it appears in the Montreal-Cambridge-Tololo survey of southern subluminous blue stars in the south Galactic cap [Lam00].

The sky and extinction corrected light curve (Fig. 6.11, upper panel) shows continous presence of cirrus clouds troughout the half hour observation run. The differential photometry is adequate, and the corresponding amplitude spectrum gives an average level of 1.1 mma for the differential photometry versus the first reference star and 0.98 versus the second. The highest peak in the spectrum, at 4.8 mHz, is not significant at 3.0 mma. It is persistent, although at an amplitude of only 2.1 mma, in the amplitude spectrum corresponding to the differential photometry between the target and the second reference star, and vanish completely in the spectrum of the relative photometry between the two reference stars. This might be interpreted as a hint of a weak pulsation, although by no means conclusive. It would be interresting to check this target again to see if the period is real or not.

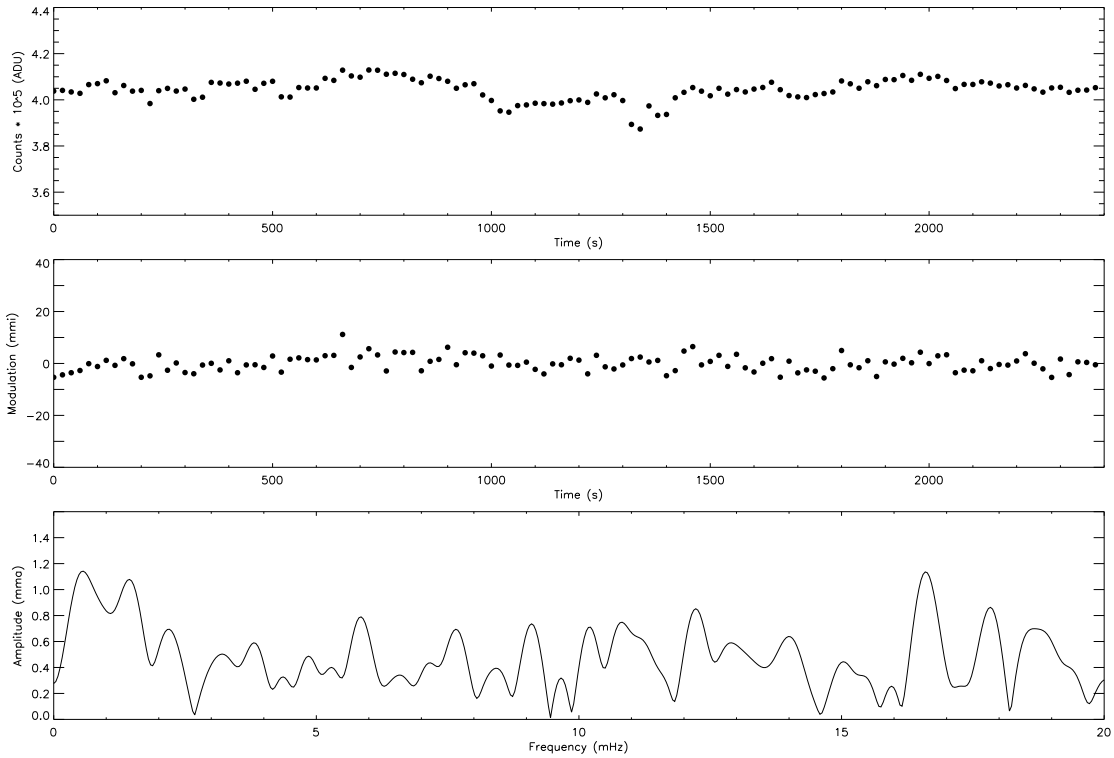


Figure 6.12: Light curve, differential photometry and amplitude spectrum for HE 0324−2529.

6.5.3 HE 0324−2529

This light curve shows some slight extinction from cirrus activity troughout the run. The clouds induces variations in the seeing that cause some light in the extended wings of the stellar image on the CCD to fall outside the aperture, when a small aperture is chosen. Altough barely noticeable at only a few milli modulation intensities (mmi) this effect shows up as a noticeable $1/f$ component in the amplitude spectrum. Increasing the aperture reduces this effect considerably, and in the light curves and amplitude spectrum shown in Fig. 6.12, which have been processed using an aperture radius of 17 pixels, the peaks below 5 mHz in the amplitude spectrum are half the amplitude of those in the corresponding specrum produced with an aperture of 12 pixels. The average level in the rest of the amplitude specrum is also reduced, although not that dramatically. There are no significant peaks in the amplitude specrum.

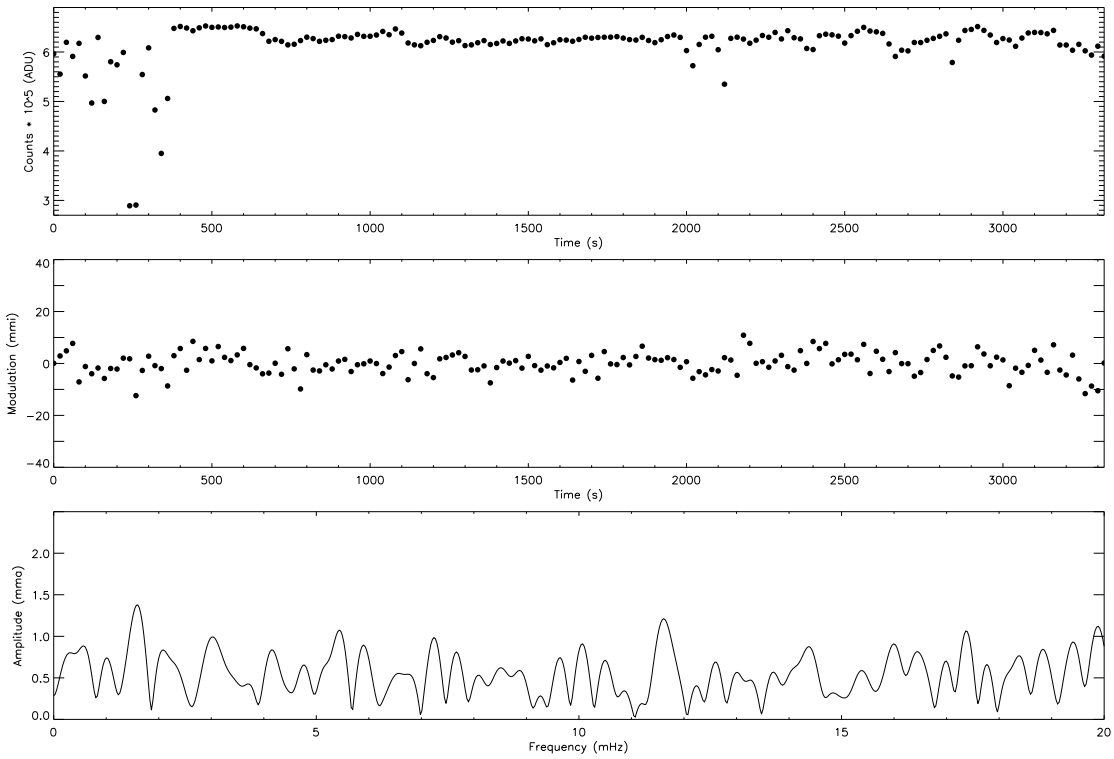


Figure 6.13: Light curve, differential photometry and amplitude spectrum for HE 0405−1719.

6.5.4 HE 0405−1719

In this run we see some cirrus activity that create quite significant extinction in the first 200 seconds of the run, and on a few occations later, but the differential photometry appears good. The mean noise level in the amplitude spectrum is acceptable at a level of 0.55 mma. There are no indications of any significant pulsations in any of the amplitude spectra for this target.

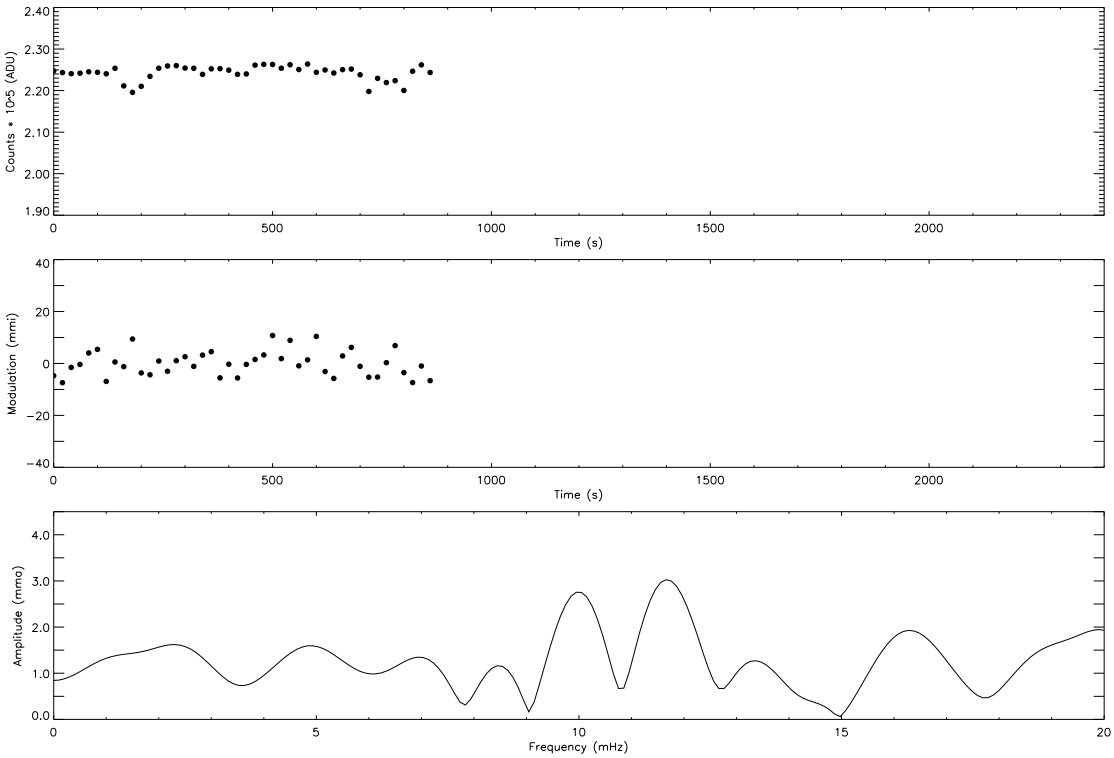


Figure 6.14: Light curve, differential photometry and amplitude spectrum for HE 0429−2448.

6.5.5 HE 0429−2448

This is a very short observation series (15 minutes), so the resolution of the amplitude spectrum is very poor. The observations were terminated early, since the light curve showed no sign of any variability, and we were at the last hour before twillight on the final night of our observation run.

The noise level in the amplitude spectrum is poor, just below 1.3 mma. The features at 10 and 12 mHz dissappear when the differential photometry is plotted towards the second reference star, but this is more noisy. When all three reference stars are taken together the noise drops to 1.2 mma, and the two peaks drops to 2.2 and 2.8 mma. Although the pair of peaks look suspicious they are not significant, but we cannot exclude low level pulsations on the basis of this short light curve. No indications of any pulsations in the relevant frequency region can bee seen in the amplitude spectrum.

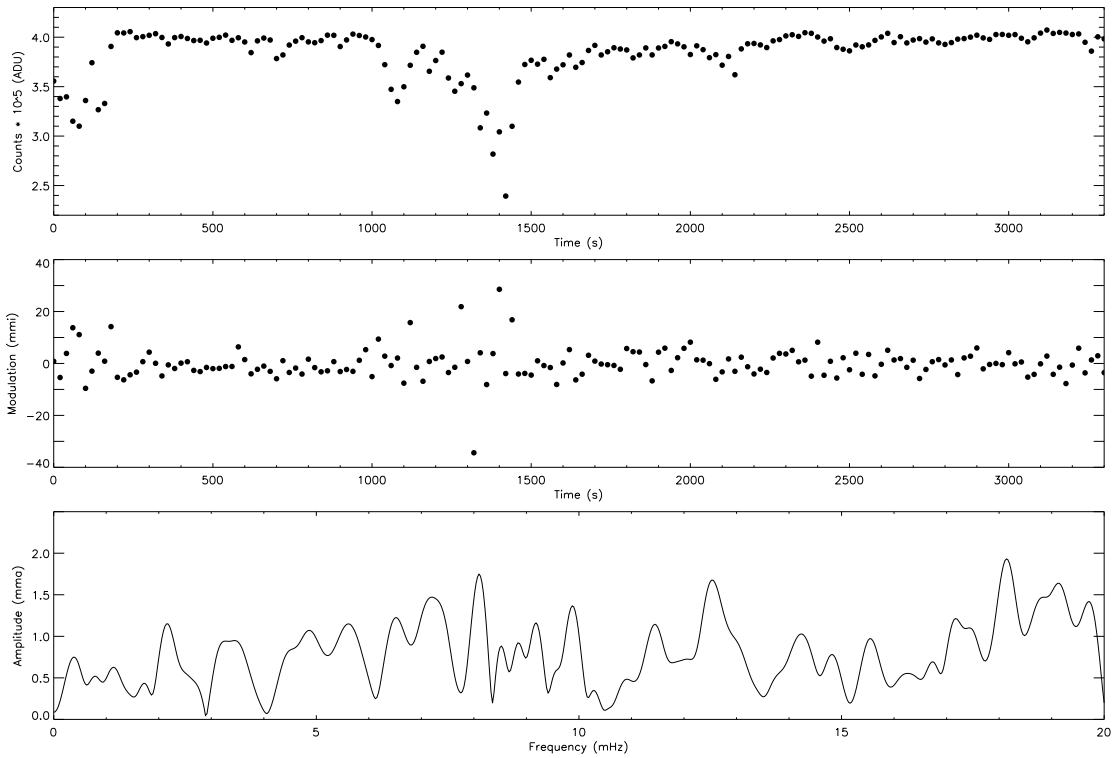


Figure 6.15: Light curve, differential photometry and amplitude spectrum for HE 2205−1952.

6.5.6 HE 2205−1952

The differential photometry light curve (Fig. 6.15, middle panel) is clearly more noisy in the parts where extinction by clouds appear. The differential photometry technique seems to do a poorer job in this case than in other observations with comparable run lenghts. The average level in the amplitude spectrum of the differential photometry (lower panel) is about 0.80 mma, and is brought down to 0.70 mma if the sum of both reference stars are used for the differential photometry. No indications of any significant pulsations can be seen.

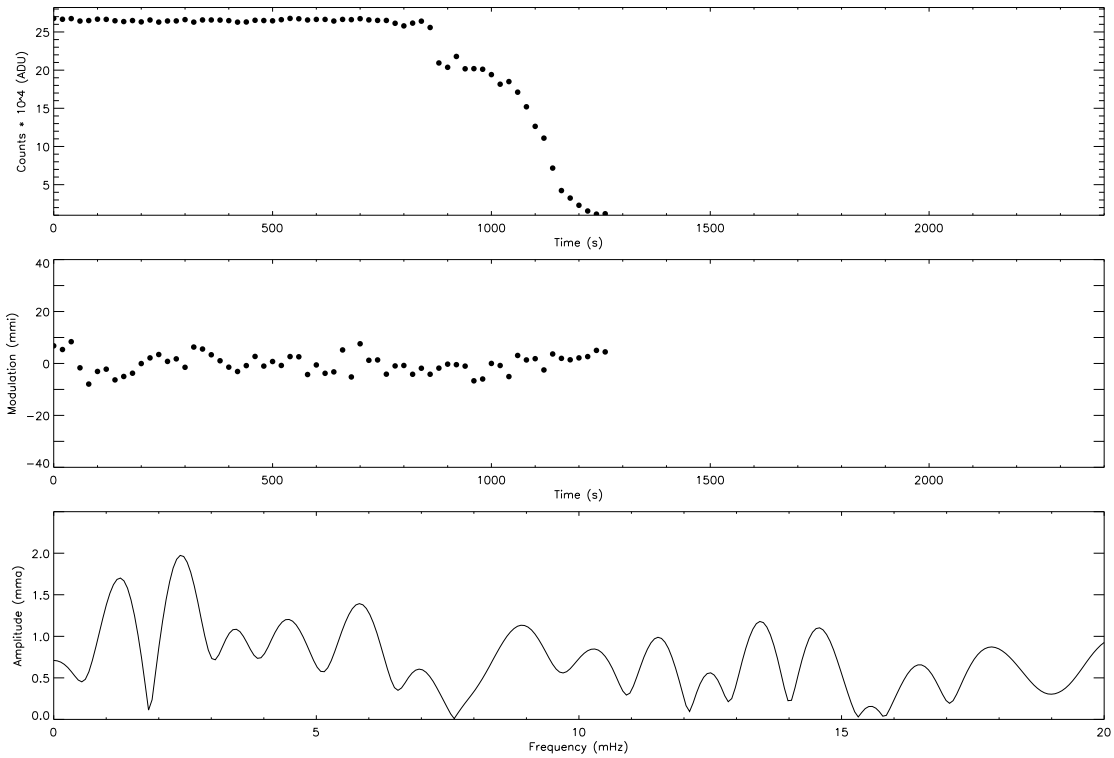


Figure 6.16: Light curve, differential photometry and amplitude spectrum for HS 0023+3049.

6.5.7 HS 0023+3049

This light curve was interrupted after about 20 minutes due to a rapid onset of thick clouds, so the resolution of the amplitude spectrum is quite poor. At an average noise level of 0.73 mma, it is still sufficiently good to establish that there are no significant pulsations in this target in the frequency range that we are interrested in here.

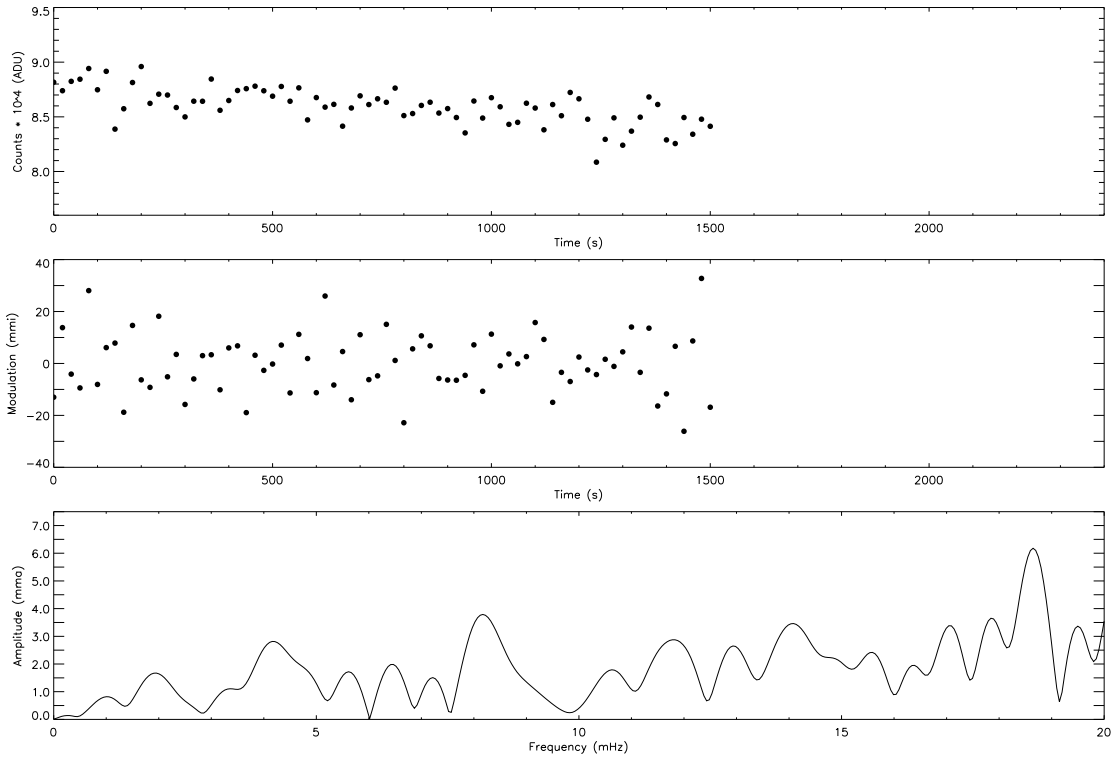


Figure 6.17: Light curve, differential photometry and amplitude spectrum for HS 0035+3034.

6.5.8 HS 0035+3034

This is another of the short lightcurves made during the rush of the last two hours of our observation run. The differential light curve (middle panel of Fig. 6.17, above) displays significant high frequency noise, as can also be seen in the amplitude spectrum. Processing with different aperture radii shows large variations in the resulting noise in the amplitude spectrum, increasing with increasing aperture size. This indicates that the noise is from the sky background, and this is likely since the target is faint, only $B = 15.8$, and observed at a zenith angle of 60 the airmass is around 2. Also, all three reference stars are significantly fainter (roughly $B = 17.2$), so significant background noise is not surprising.

The average of the amplitude spectrum is 1.8 mma, but as can be seen from the amplitude spectrum the noise is slanted towards the high frequencies. If we take the mean level between 0 and 7000 mHz, as an indicator of the noise we get 1.1, which makes the peak at 8.2 mHz barely significant with an amplitude of 3.8. Using the differential photometry with respect to one of the other reference stars or all three together gives peaks of similar amplitude at other places in the spectrum, so we do not give any significance to the 8.2 mHz feature.

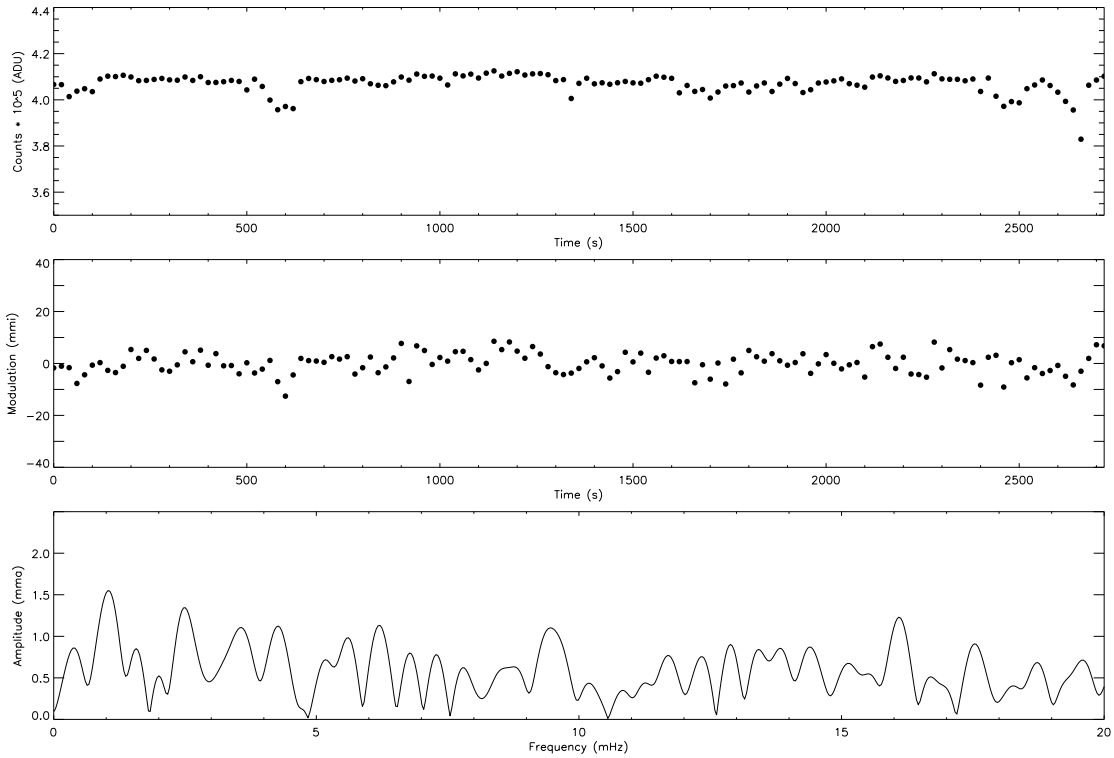


Figure 6.18: Light curve, differential photometry and amplitude spectrum for HS 0048+0026.

6.5.9 HS 0048+0026

This star is also known as PHL 860 and PG 0048+004 in the Palomar–Green catalog of ultraviolet-excess stellar objects [Gre86], and it also appears in the catalog of spectroscopically identified hot subdwarf stars from the South African Astronomical Observatory [Kil88].

This target was observed without any filter. The sky subtracted light curve (upper panel of Fig. 6.18) shows some interruptions by clouds. When this data set was processed using an aperture radius of 12 pixels, these interruptions could still be identified in the differential light curve. The corresponding amplitude spectrum showed a significant $1/f$ component with a peak at 1 mHz of 3.5 mma. The data were reprocessed with increasing aperture radius, and the noise dropped significantly with increasing radius, although the average gain in signal when going from 12 to 19 pixels was only of the order of 20 percent. The light curves and amplitude spectrum showed in Fig. 6.18 was produced using an aperture radius of 17 pixels, and a mean amplitude spectrum level of 0.60. No significant pulsations are evident from the amplitude spectrum.

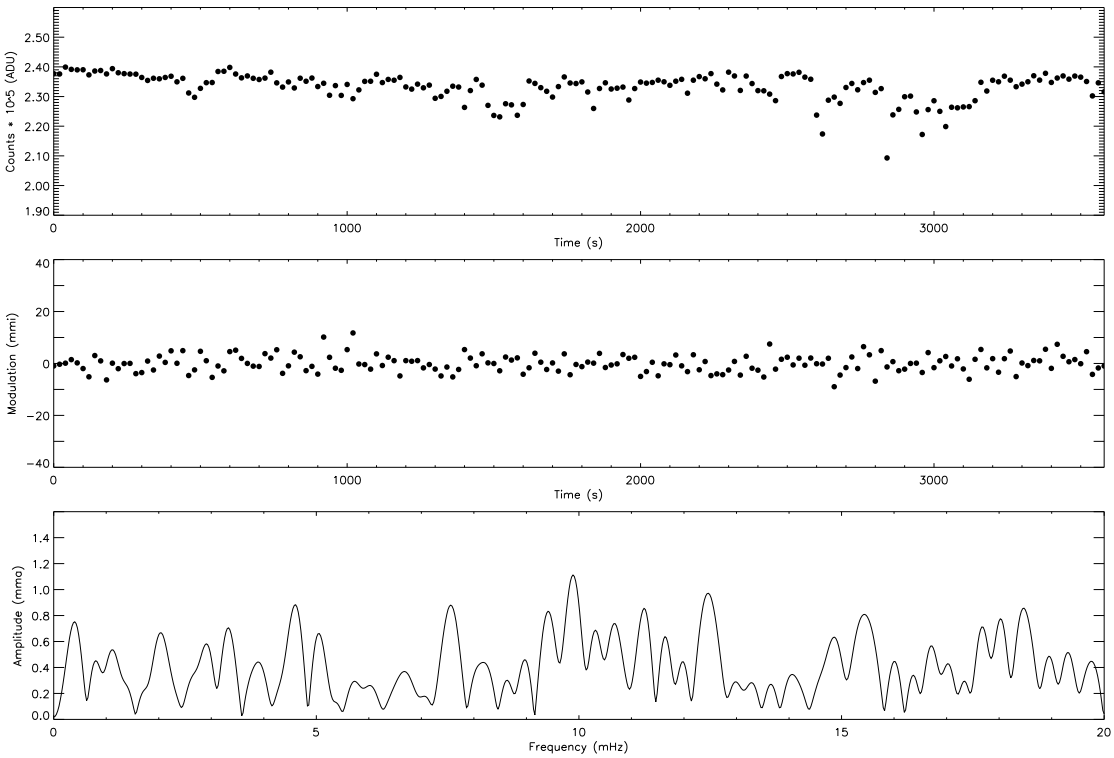


Figure 6.19: Light curve, differential photometry and amplitude spectrum for HS 0055+0138.

6.5.10 HS 0055+0138

As with the previous star HS 0055+0138 also appears in the Palomar–Green catalog (as PG 0055+016), and was included in the study of Kilkenny *et al.* [Kil88].

Unfortunately, the reference star in this frame sequence happened to fall right upon one of the serious "droplet" defects on the CCD chip surface, which contributes some noise to the reference star data. The noise is still very low, although the amplitude spectrum shows a peculiar broad feature around 10 mHz, probably due to the stars movements in and out of the low sensitivity region of the chip. We have only one reference star for this target, so we have no way of cross checking these results. There is also a significant ammount of cirrus activity, but still we get a noise level as low as 0.41 mma. Considering the resulting amplitude spectrum (Fig. 6.19, lower panel) it is safe to conclude that we have no significant pulsations in this target, in spite of the problems with the raw data.

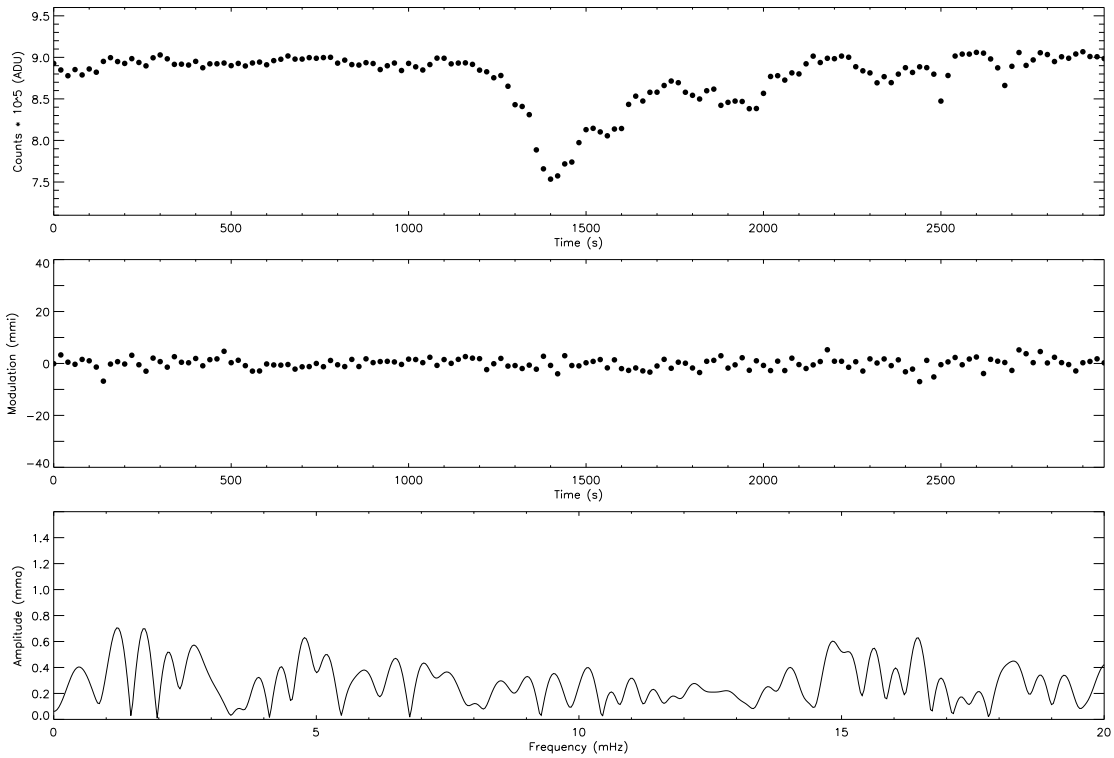


Figure 6.20: Light curve, differential photometry and amplitude spectrum for HS 0213+2329.

6.5.11 HS 0213+2329

The differential photomtetry of this target is very stable, despite the cirrus that can be seen creating a significant dip in the middle of the series. The amplitude spectrum is down at an impressive average of 0.28, and no significant pulsations can be seen.

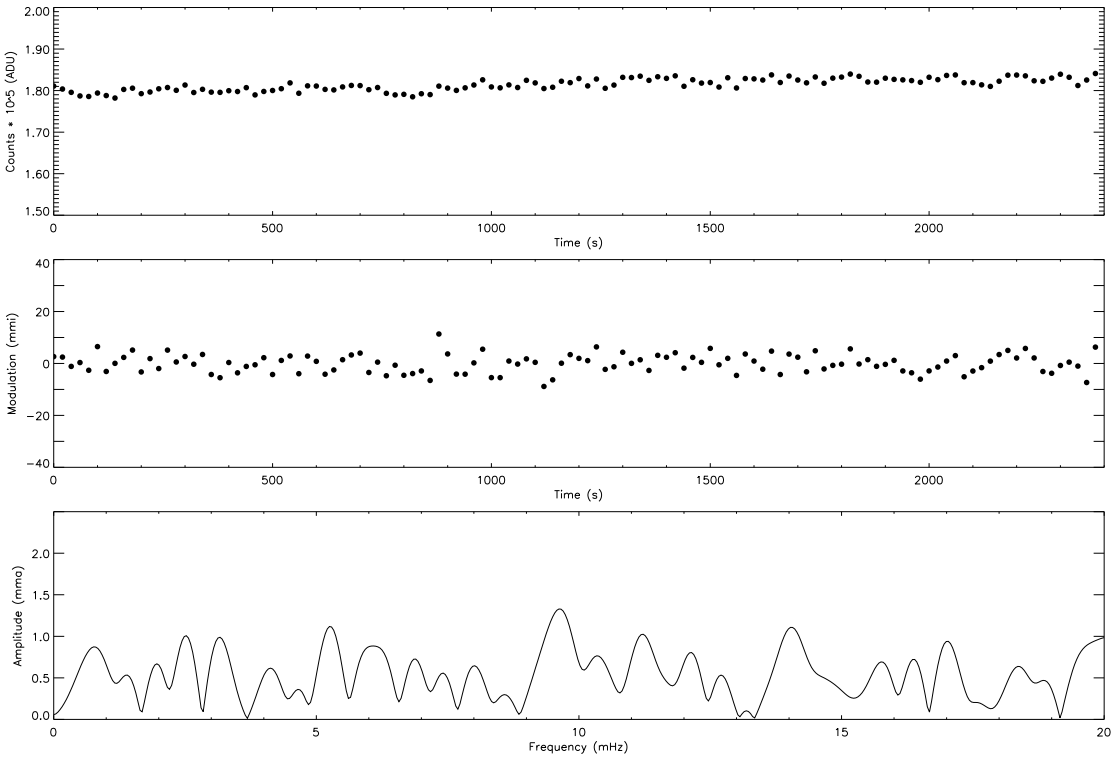


Figure 6.21: Light curve, differential photometry and amplitude spectrum for HS 0232+3155.

6.5.12 HS 0232+3155

Another unproblematic light curve with amplitude spectrum noise level at 0.50 mma for the differential photometry relative to both reference stars, and no significant pulsations.

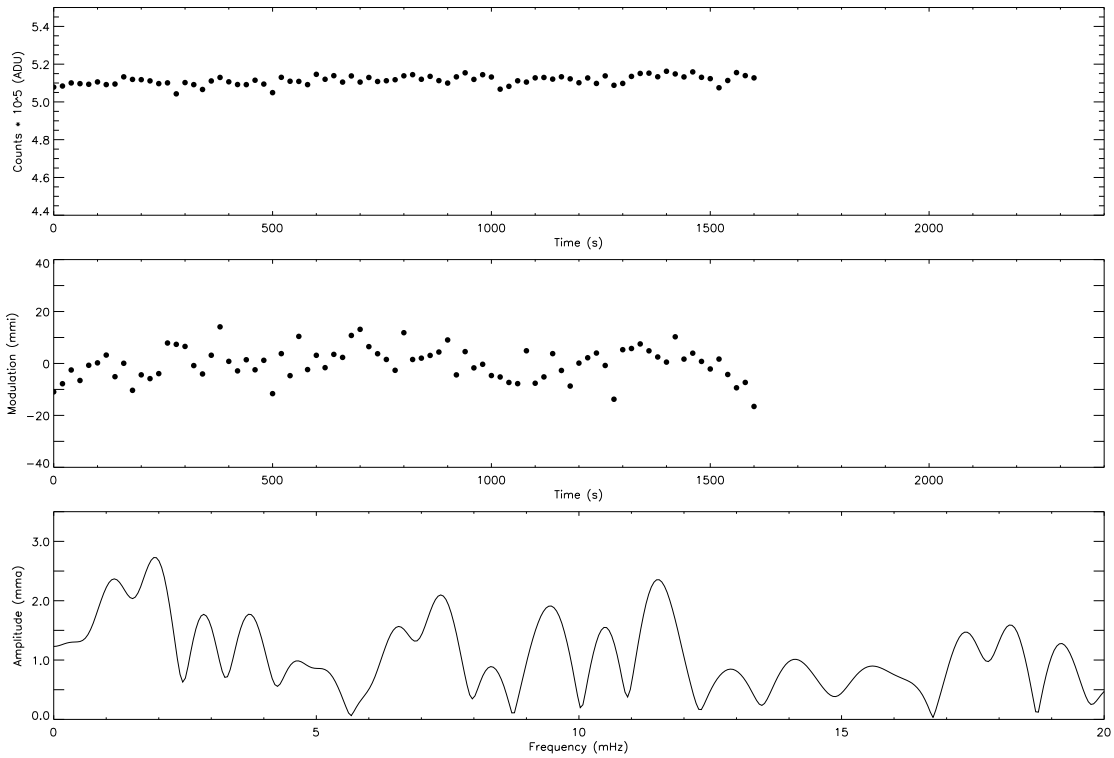


Figure 6.22: Light curve, differential photometry and amplitude spectrum for HS 0546+8009.

6.5.13 HS 0546+8009

The primary reference star has happened to fall upon a particularly bad flat field error on the CCD surface, which completely destroys the photometry. The noise level is more than 5 mma in the amplitude spectrum between the target and this reference star. Fortunately, we have another reference star, which, although it is 1.8 magnitudes fainter than the target, gives a quite acceptable differential photometry. Thus, the differential photometry and amplitude spectrum shown in Fig. 6.22 is that of target relative to this second reference star. Altough not among the best with a mean amplitude 1.1 mma, it is adequate for such a short light curve.

Actually, this is one of the few cases where the raw photomtery is so good (completely without interference from clouds) that the photometry from the raw data performs better than the differential method (due to the problems with the reference stars). The average of the amplitude spectrum of the extinction corrected sky subtracted light curve of the target alone is only 0.60 mma. Still without any significant peaks.

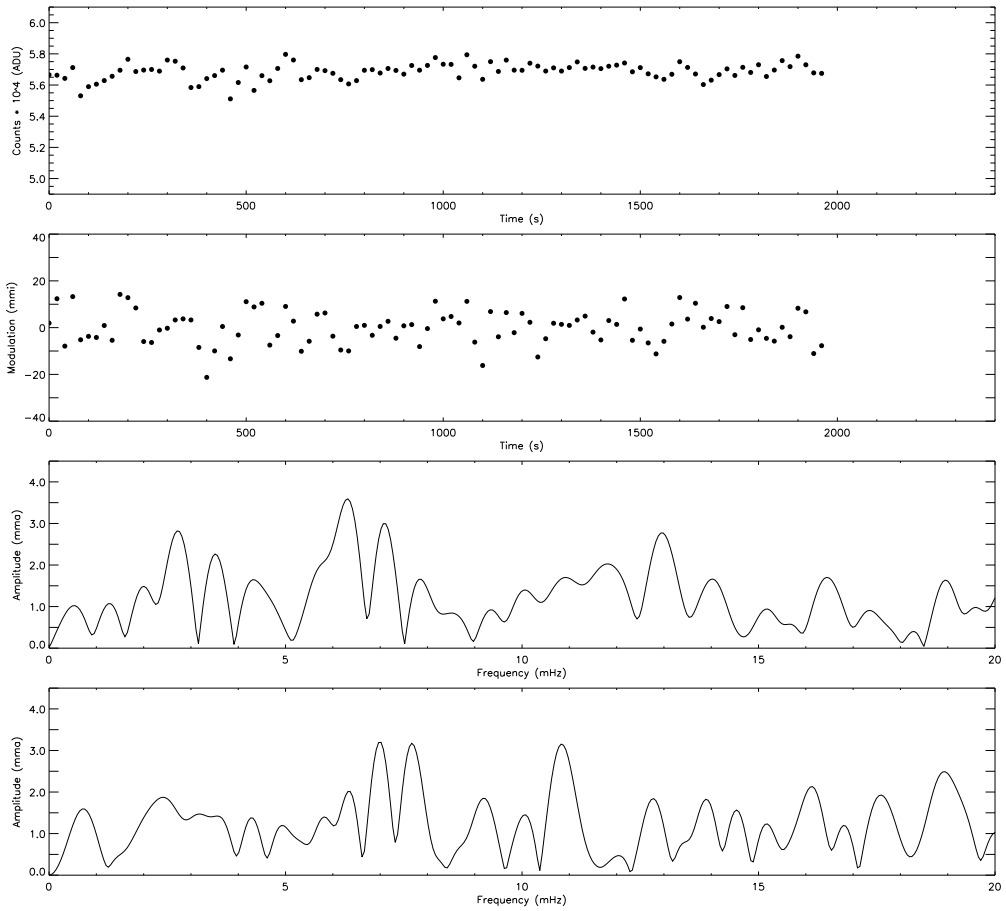


Figure 6.23: Light curve, differential photometry and amplitude spectrum for HS 0600+6602. The lower panel shows the amplitude spectrum of the target light curve alone.

6.5.14 HS 0600+6602

This light curve appears at first look to be quite stable and the amplitude spectrum of the differential photometry has a mean noise level of 1.2 mma relative to both reference stars, with no peaks above the three sigma level. But the highest peak in the amplitude spectrum at 6.4 mHz is persistent at around 3.6 mma in the spectrum of the differential photometry of the target relative to both reference stars, while it vanishes completely in the spectrum of the differential light curve between the two reference stars. Altough a 2.5 σ peak is by no means significant, the persistance of the peak togeter with the fact that the peak is in the expected frequency range for sdB pulsators makes this target interresting for a follow-up study.

Since the light curve of the target is very stable (no cirrus activity), it is instructive to consider the FT of the target alone. It is shown in the fourth panel of Fig. 6.23, and we see that the peak at 6.4 mHz is significantly lower here, and that the peaks at

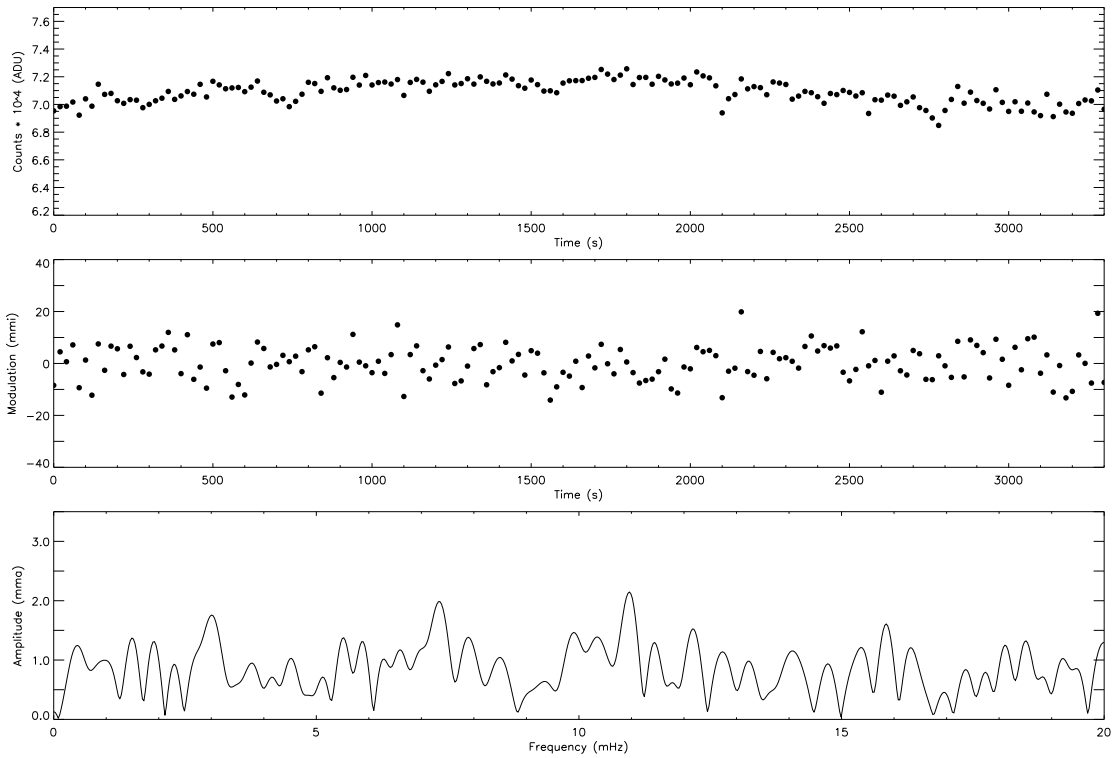


Figure 6.24: Light curve, differential photometry and amplitude spectrum for HS 2151+0214.

7.0 and 7.8 mHz is dominating at around 3.2 mma. The mean level in this amplitude spectrum is also 1.2 mma, so there is a possibility that this is a low level pulsator with multiple periods in the 6 to 8 mHz range, but only a much longer light curve can resolve this question. I would suggest that this target is flagged for a follow-up study on our next search run.

6.5.15 HS 2151+0214

This light curve is quite clean with a mean amplitude level below 0.86 mma both for the differential photometry between the target and first reference star, and below 0.80 for the target and both reference stars together (shown in the lower panel of Fig. 6.24). The peak in the amplitude spectrum at around 7 mHz is almost significant at an amplitude of 2.0 mma for the differential photometry between target and the first reference star, but it is not present in the amplitude spectrum between the target and second reference star. The peak at 11 mHz is present at 2.0 mma in the amplitude spectra for the differential photometry versus either reference star (and the sum, of course).

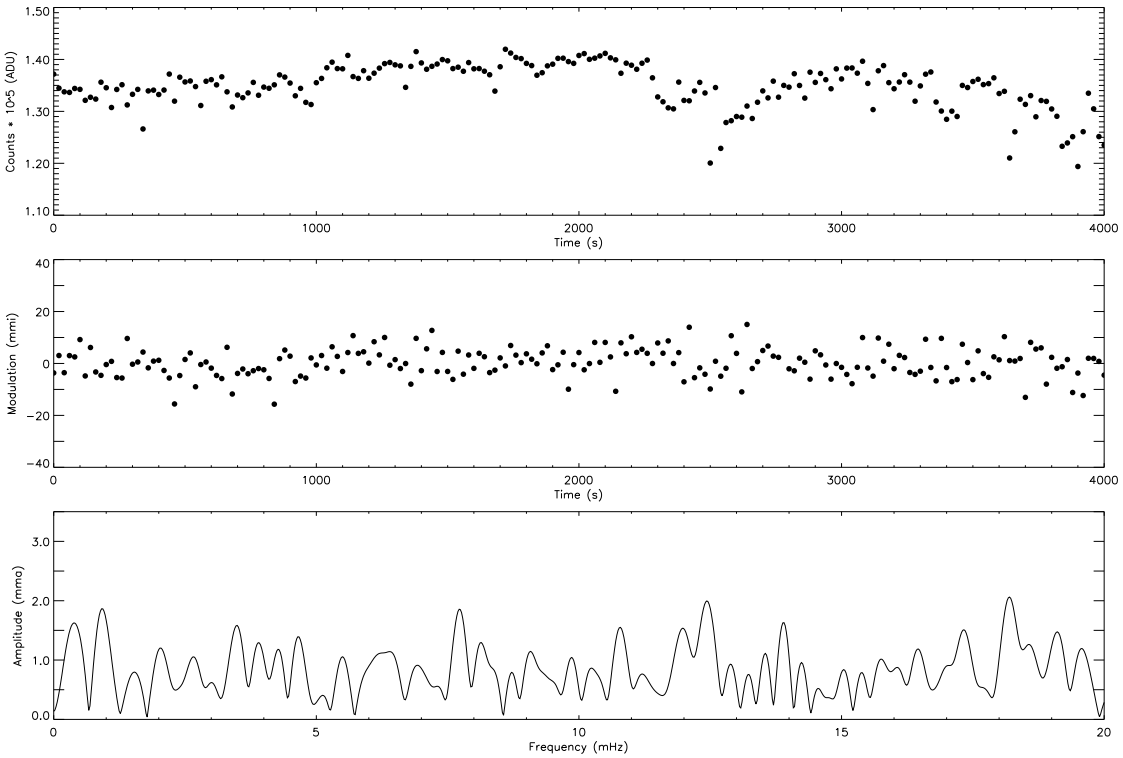


Figure 6.25: Light curve, differential photometry and amplitude spectrum for HS 2156+2215.

6.5.16 HS 2156+2215

Altough some slight presence of cirrus is evident from the sky subtracted light curve (Fig. 6.25, upper panel) the extinction corrected differential photomtery (middle panel) is of good quality. Again, the mean of the amplitude spectrum is between 0.70 and 0.90 mma for the various differetial light curves, and as low as 0.57 mma for the differential photometry versus the sum of all three available reference stars (shown in the lower panel of Fig. 6.25). No significant peaks are found, although the peak at 12.5 mHz comes close to 3 sigma with 1.7 mma.

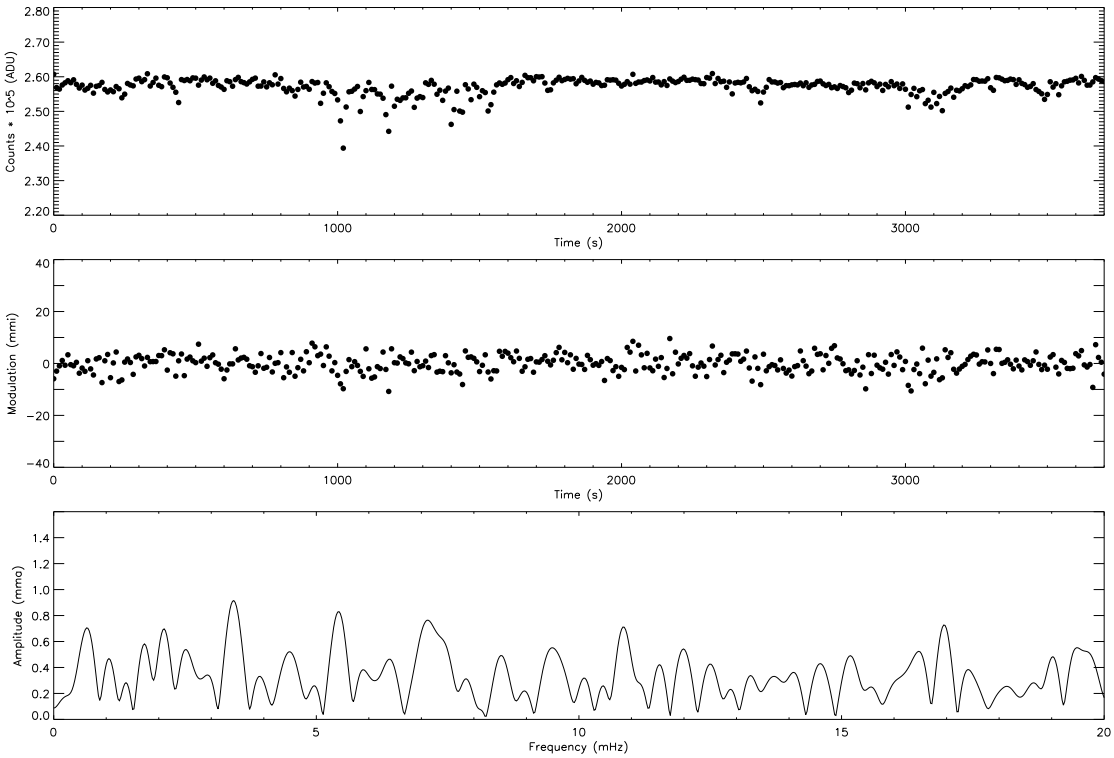


Figure 6.26: Light curve, differential photometry and amplitude spectrum for HS 2206+2847.

6.5.17 HS 2206+2847

This target caused some considerable headache because the extinction corrected light curve did not look anywhere close to flat. It turned out, when examining the FITS headers in the original CCD data frames that the observations have accidentaly been made without any filter. When the light curve was processed with a small extinction coefficient ($K = 0.05$) the result looks much better, as can be seen in the upper panel of Fig. 6.26.

Again we see some interruptions by passing cirrus clouds in the light curve of the target which are effectively removed in the differential photometry. The noise level is very low; below 0.35 mma for the amplitude spectra of all the various differetial light curves. This target was reobserved after observations from the first search programme run showed some suspicious behavior. But, as we can see from the new observations, there are no indications of any pulsations.

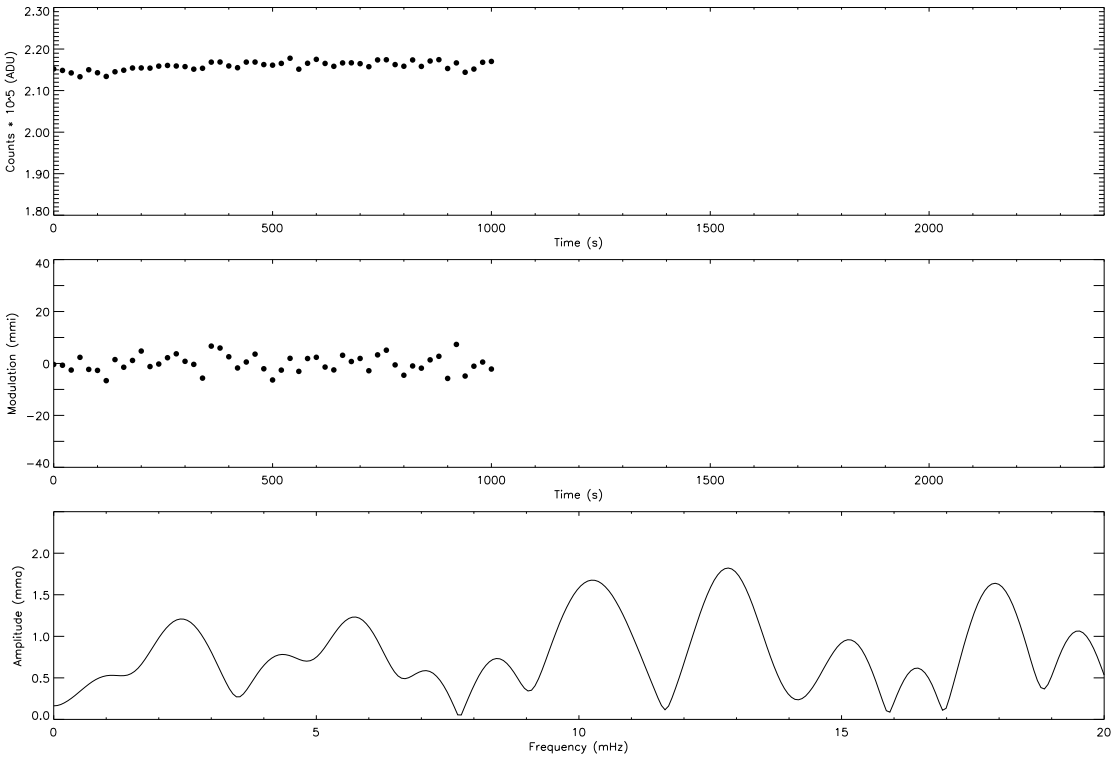


Figure 6.27: Light curve, differential photometry and amplitude spectrum for HS 2208+2718.

6.5.18 HS 2208+2718

A flat field error is present close to the first reference star, which is also more than two magnitudes brighter than the target. The amplitude spectrum of the differential photometry between target and this reference star has a mean of 1.05 mma. The second reference star shows no sign of any errors and is also much closer in brightness to the target (only 0.9 magnitudes brighter). The extinction corrected differential light curve and corresponding amplitude spectrum shown in the middle and lower panels of Fig. 6.27 are that of the target and second reference star, and the amplitude spectrum have a mean level of 0.79 mma.

Altough the light curve is only 1000 seconds long, there are no indications of any significant variability in this object. The light curve is to short to completely rule out variability at the 1 mma level.

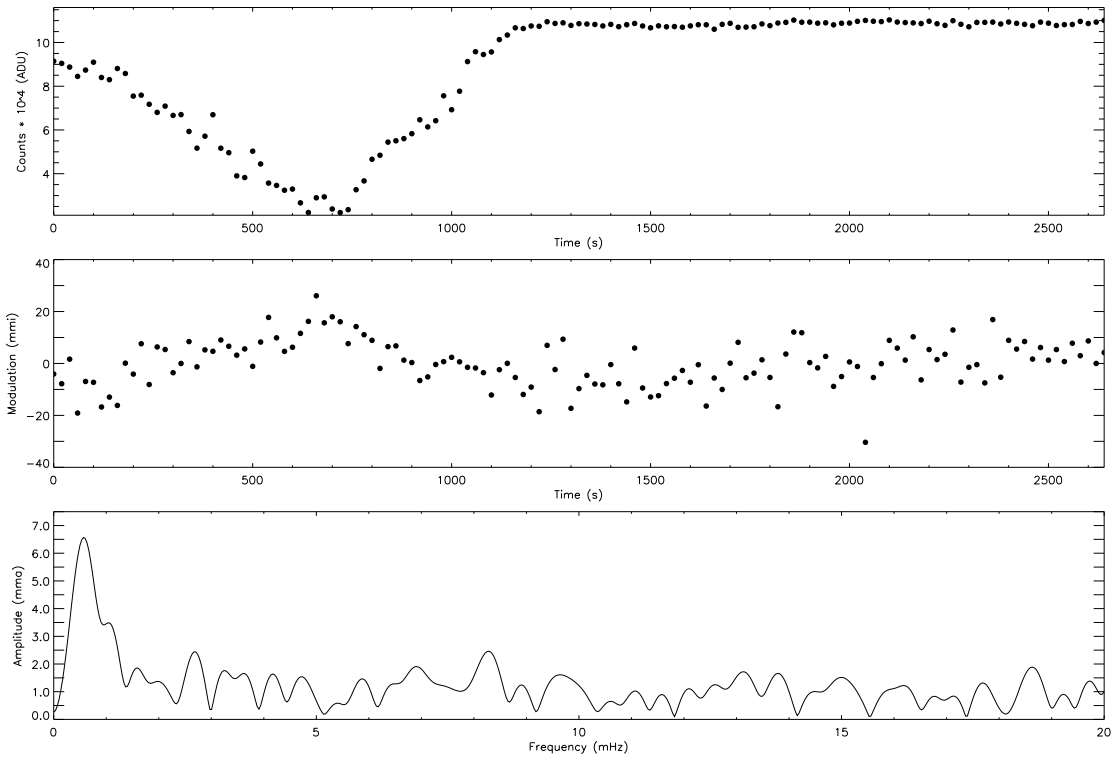


Figure 6.28: Light curve, differential photometry and amplitude spectrum for HS 2209+2840.

6.5.19 HS 2209+2840

This light curve shows a very dramatic (up to 80 percent) extinction by clouds in the first part of the extinction corrected sky subtracted data (upper part of Fig. 6.28). The differential photometry removes most of the effects from this, but not completely, as is evident from the small bump at around 700 seconds in the middle panel of the above figure. Also, the amplitude spectrum (lower panel) gets punished from this incomplete correction, showing a significant peak in the low frequency end of the spectrum. The remaining part of the spectrum shows no significant peaks, and despite the dramatic event in the raw light curve we get an acceptable noise level in the frequency range above 3 mHz; 1.04 mma.

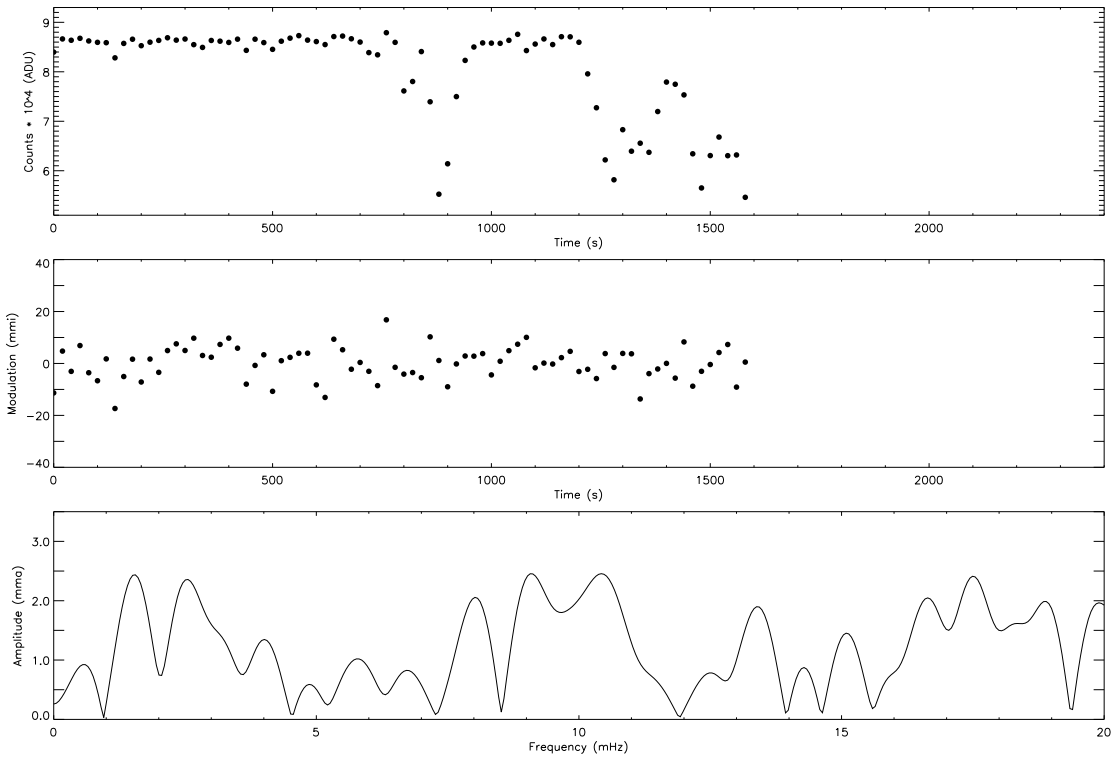


Figure 6.29: Light curve, differential photometry and amplitude spectrum for HS 2225+2220.

6.5.20 HS 2225+2220

This light curve also show some significant extinction by clouds towards the end of this rather short run. The differential photometry appearantly does a good job in removing the effects. The light curve is short, so we do not get the noise level much below 1.2 mma. The normalised differential photometry and corresponding amplitude spectrum shown in Fig. 6.28 is that of the target versus the sum of both reference stars. There are no indications of any pulsations.

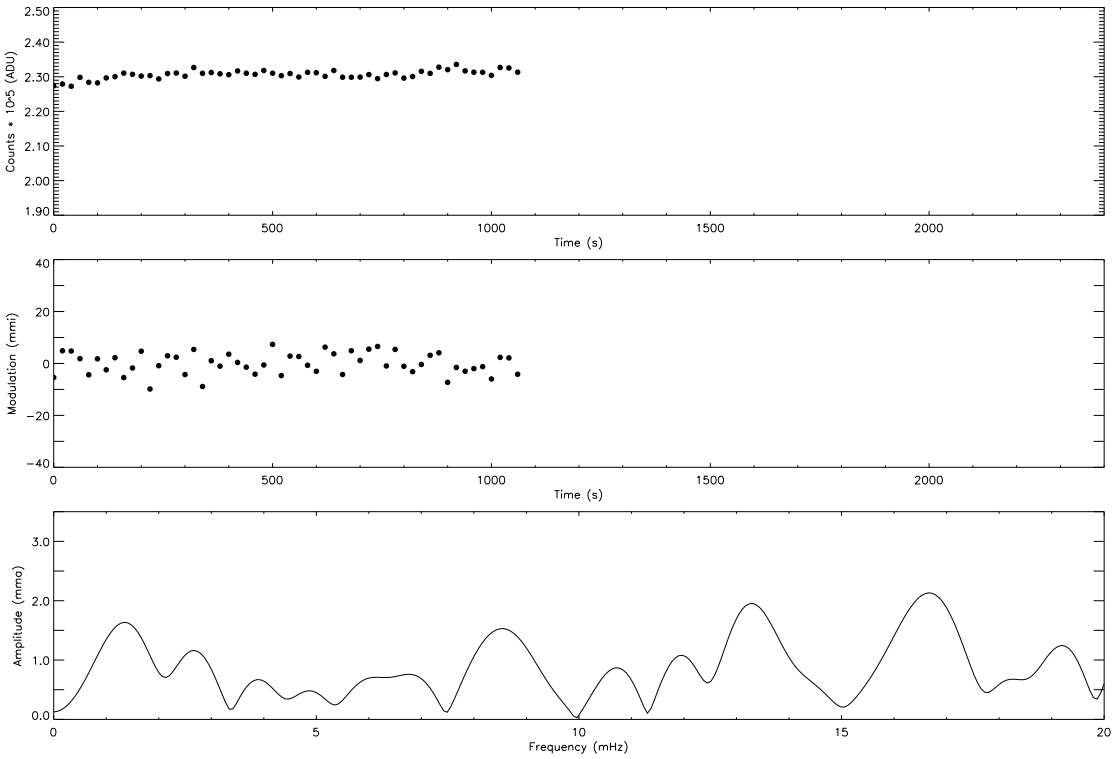


Figure 6.30: Light curve, differential photometry and amplitude spectrum for HS 2229+0910.

6.5.21 HS 2229+0910

This is another short light curve. Observations were terminated after only about 1000 seconds with a mean noise level of about 0.86 mma.

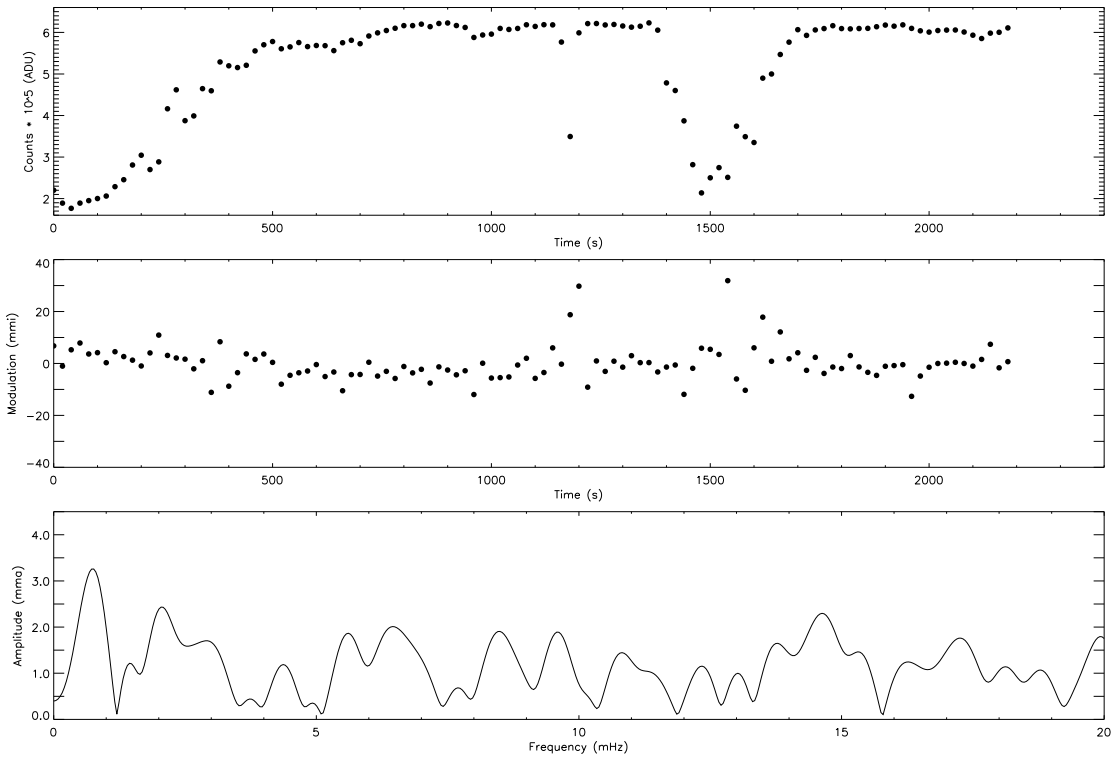


Figure 6.31: Light curve, differential photometry and amplitude spectrum for HS 2242+3206.

6.5.22 HS 2242+3206

This light curve started out with significant extinction by clouds and was disturbed by clouds one more time, at around 1500 seconds. The whole light curve is more noisy than what is seen in runs of comparable length; the mean level in the amplitude spectrum is just below 1.2. No indication of any pulsations can be seen, but as we can see from the amplitude spectrum, pulsations at the 2 mma level cannot be ruled out. This target should be reobserved.

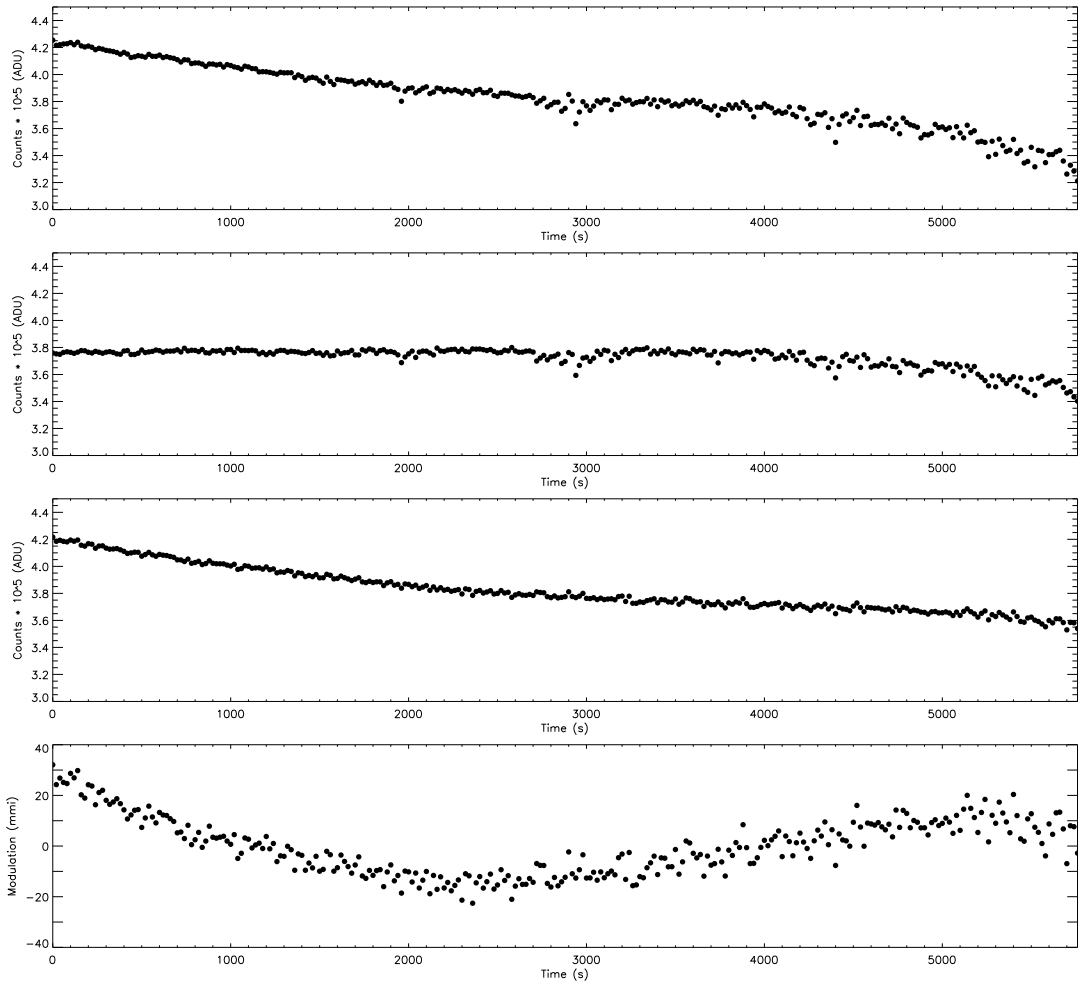


Figure 6.32: Extinction corrected light curve for the target and reference star, and the differential photometry before and after normalization for HS 2333+3927.

6.5.23 HS 2333+3927

For this target we show both the extinction corrected light curve of the target and the first reference star (Fig. 6.32, upmost and second upmost panel, respectively). The second lowest panel shows the extinction corrected differential photometry, and the lowest panel shows the result of applying the normalization algorithm to that differential light curve. This shows a slow, apparently periodic, variation in the differential photometry, but as can be seen when the figures are taken together, this is just an artifact of subtracting a straight line through the differential photometry. The amplitude spectrum is not shown, because it reveals nothing but the obvious long term variation. A higher order polynomial fit must be applied to get any useful higher frequency components, but this has not been implemented yet.

Three reference stars were observed during this run, so it should be easy to determine

the origin of these variations. During this run the zenith angle grows from 44 to 62 degrees, and it appears as the target was setting into dust or clouds, because the extinction correction fails for the last quarter of the light curve.

All four (sky subtracted) light curves has the same overall shape before extinction correction as that of the target seen in the upper panel of Fig. 6.32, above. After running `rtcorr` the overall shape of the first and third reference stars are flat, while that of the second reference star (Ch. 3), which is also the brightest, starts to drop of at around 4000 seconds into the almost 6000 second long run. Running `rtcnv` reveals that all targets get severely elongated toward the end of the run, and that the star in Ch. 3, which is not very well centered, starts to suffer vigneting from around 2/3 into the run, so we should not use this star as a reference. Also the first reference star (Ch. 2) can be seen from its extinction corrected light curve to suffer from some vigneting, but only from about 5000 seconds and onwards, and not enough to explain the peculiar ups and downs in the extinction corrected differential photometry between the target and the first reference star, shown in the middle panel of Fig. 6.32. The differential photometry between the first and third reference star (not shown) is convincingly flat, execpt for the last 1500 seconds of the light curve where the vignetting starts, and furthermore the differential photometry between the target and the fourth reference star shows the same behaviour as that of the target and Ch. 2, shown in the middle panel.

Since vigneting only becomes significant in the last third of the light curve, this effect cannot explain the variations seen in the lower panel of Fig. 6.32. A colour effect due to the very different colours of the hot blue target and the (most likely) red reference stars is unlikely since the observations are filtered, but can not be ruled out completely. If the colour effect could be of this level we would have expected to see similar behaviour in other targets, which we have not. Thus, although dubious, the feature can be intrinsic to the star, possibly a reflection effect if the star is in a binary system. Longer observation runs must be undertaken before a reliable analysis can be undertaken.

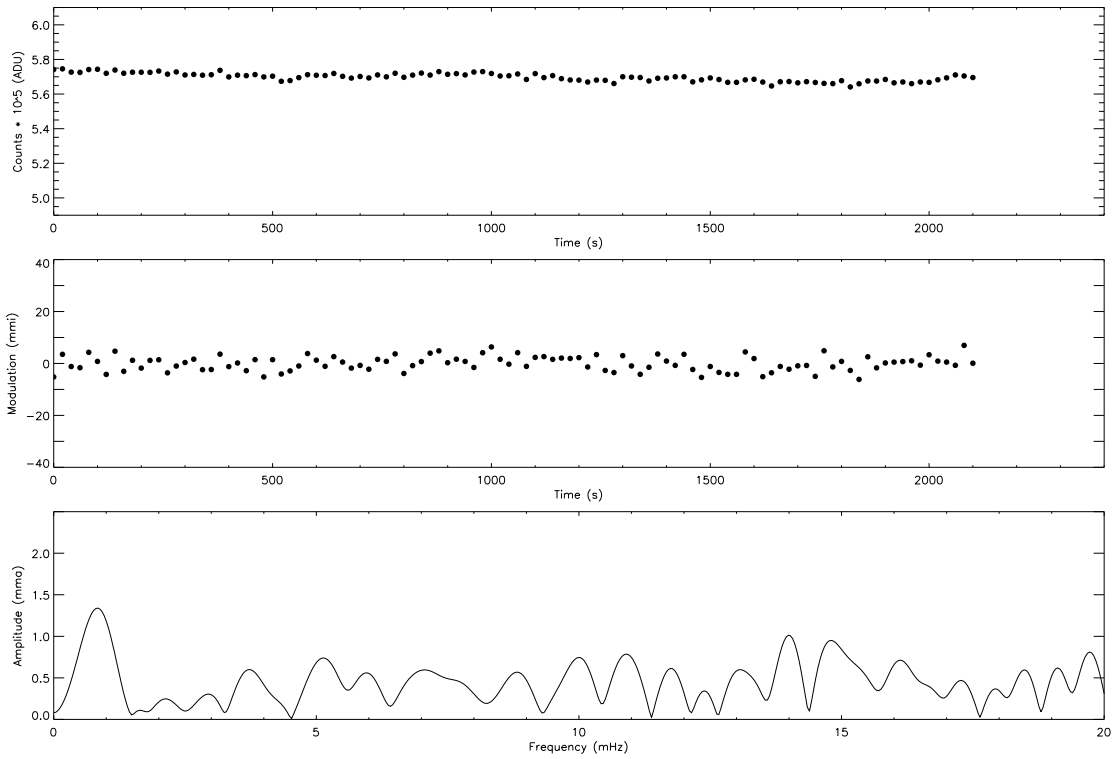


Figure 6.33: Light curve, differential photometry and amplitude spectrum for PB 6740.

6.5.24 PB 6740

PB 6740 is also listed in the Palomar–Green catalog as PG 0209+017 [Gre86] and appeared in the South African subdwarf catalog [Kil88].

This light curve is as uneventful as can be. The average level of the amplitude spectrum is less than 0.5 mma, and there are no signs of any pulsations.

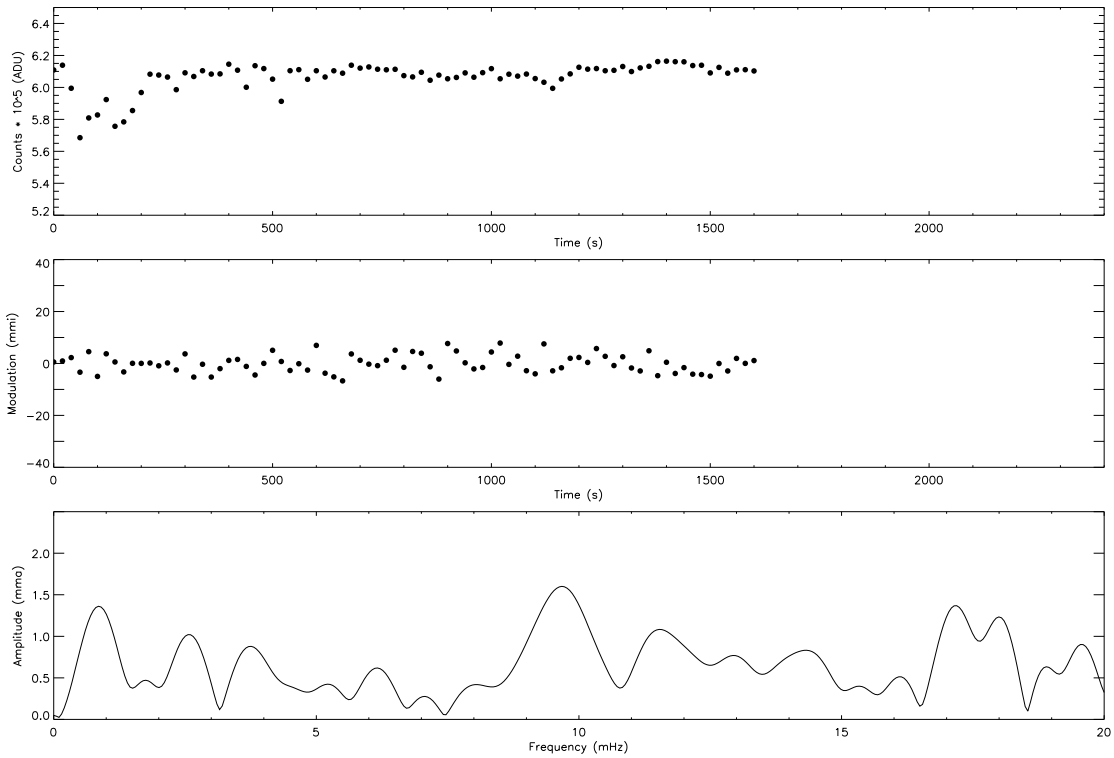


Figure 6.34: Light curve, differential photometry and amplitude spectrum for PG 2233+1418.

6.5.25 PG 2233+1418

Another uneventful light curve. The average level in the amplitude spectrum is about 0.65 mma. The peak at 9.5 mHz is not significant.

6.5.26 PG 2240+105

PG 2240+105 (Green *et al.* [Gre86]) appears also in the South African subdwarf catalog [Kil88].

The raw data is not very stable, and the resulting differential photometry is also somewhat noisy. This results in an amplitude spectrum that is quite noisy, with a mean level of about 1.7 mma, but with most of the power in the high frequency end. The light curve is too short and noisy to put any limits on pulsations. say anything about pulsations below the 3 mma level.

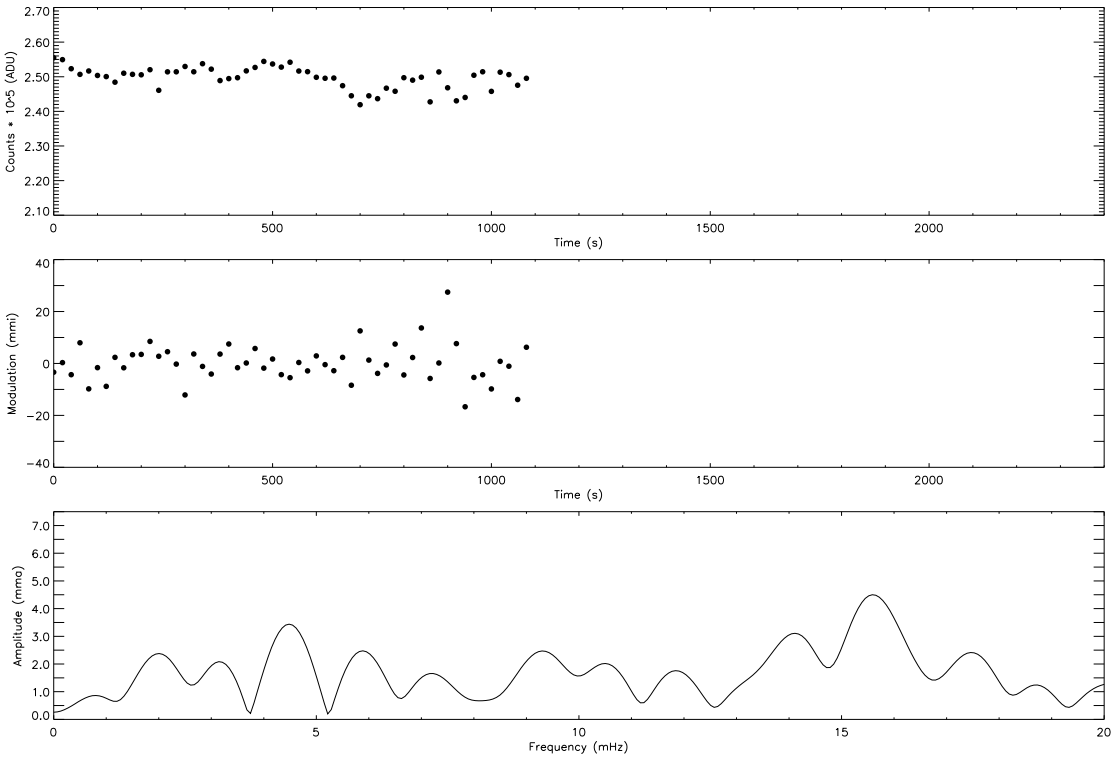


Figure 6.35: Light curve, differential photometry and amplitude spectrum for PG 2240+105.

6.6 It's a Wrap

Our first variable star search campaign using the windowed CCD photometry technique proves that this system is excellent for this kind of work, and gives several improvements compared to the standard photoelectric technique. With a medium size telescope we can detect low amplitude pulsators in as little as 30 minutes, unless the pulsations are suppressed due to beating, and get reliable identification of the primary modes in one hour.

Chapter 7

The Road Ahead

This thesis work has presented a working solution for time resolved photometry with modern astronomical CCD cameras. In the first chapters, the design for a new portable CCD camera was introduced, and the software for making it run and produce useful data was described. However, at some point we diverged from the testing of the Tromsø CCD Photometer, and performed the tests of the windowed readout method using the HiRAC camera on the Nordic Optical Telescope instead. The reason for this diversion is related the fact that my time was running out. The time it took to obtain the CCD controller electronics and the funding for the chip itself, delayed this project with several years, and with the tests of the software from the NOT, this thesis became more than substantial enough.

Time resolved CCD photometry is a big project, and now that we have established a solid foundation upon which we can continue to investigate, many good results are sure to be coming. In this effort there is room for many people; for testing and perfecting the hardware, for observing and analysing new data, and for implementing new methods to squeeze more scientifically useful information out of the data.

7.1 Challenges

The work on the actual testing and perfectioning of the Tromsø CCD Photometer, has continued in the hands of other people. Robert Kamben spent the winter 1999–2000 at Skibotn Observatory, running the camera and obtaining good photometric data on several faint targets, for his Master thesis. His work shows in an excellent way that time resolved photometry can indeed be done on faint stars in the Auroral zone. With targets including the cataclysmic variable, AM CVn (m_B=14.5), and the ZZ Ceti variable white dwarf, GD 358 ($m_B = 13.5$), Kamben managed to extract the light curves from a sky background contaminated by events of brilliant aurora. The average sky brightness during the AM CVn observations was about 17.7 magnitudes

per arcsecond, but varying from 19.2 mag/" between outbreaks and peaking at 16.2 mag/". Even in these extreme conditions, acceptable photometry was achieved. The amplitude spectrum, as computed from the differential light curve processed with the sky annulus correction procedure, revealed the expected main peak with an amplitude of about 10 mma on a white noise background of 1.5 mma, after 10 hours of observations. An impressive result for a 50 cm telescope under such conditions!

As this is written, the CCD photometer is in the workshop, undergoing improvements to the cooling system and the shutter. In the coming winter, Master student Frank Johannesen will continue testing the CCD Photometer, and using it to optimise the performance of the Skibotn telescope tracking and guiding system. One possibility that may be investigated is to use the output from the `rtp` program centering algorithm (used for automatic recentering of the aperture) to autoguide the telescope. In any case, he will interface the telescope pointing computer to the CCD user computer so that we can get target position and reliable time information into the headers of the data files.

Furthermore, Ph.D. student Jose-Miguel Gonzales-Perez will use the CCD photometry system to investigate variable nuclei stars in planetary nebulae. This may involve more complex solutions for removing the non-uniform nebula background from the light of central star, than the simple flat sky background removal procedures that have been discussed here.

7.2 New Camera Design

The prototype CCD photometer described here has proved to be a working solution. But improvements to the timing control of the system is necessary to reach the precision desired, especially for multi-site campaigns. We can achieve this if we obtain a GPS system, and interface it through an upgrade of the Optio board. From our collaborating partner, the Institute of Material Research and Applied Science of Vilnius University in Lithuania, we have received expression of interest for building the necessary electronics parts. They are also interesting in collaborating on a second generation CCD controller, that can be made both cheaper and with a better interface to the user computer, than the Copenhagen CCD controller. Reworking the CCD data transfer interface to a state of the art solution that could connect to a notebook computer, could reduce the total weight of the whole system by 8 kg, *i.e.* almost 40%, making it possible for one person to travel with the CCD photometer without problems.

7.3 Software Improvements

Many improvements to the existing software suite have been envisioned in the course of this work, but have been slashed in order to finish this thesis. Let me mention a few of the most important ones.

First of all, a graphical user interface should be made, in order to provide a more intuitive interface to the camera, and hide all the irrelevant information from the observer. The process of performing a readout test after initialising the windows, to obtain the maximum integration time allowed for a given sequence time interval, should be made automatic. Improvements to the visualisation of the results, both imaging, light curve and spectral data, are also important.

A number of improvements to the photometric algorithms in `rtp` could be made. PSF photometry should be implemented for cases like PG 1618+563, and for comparisons in general. Some automatic data quality checks could also be implemented. Cosmic rays are not a very big problem due to the short exposure times and small apertures that are used here. In fact, not a single one has been seen in all the data obtained so far. Still, a filter to reject extremely deviant pixels could easily be implemented. Even better would be a `qed`-like editor for removing all sorts of bad points and interpolating over the gaps.

As soon as we get telescope pointing information into the photometry headers, automatic (preliminary) extinction corrections could easily be made based on the existing programs.

The software for joining together multiple observations and compute their Fourier transforms is also needed, but for the moment we can manage by converting the output to `Quilt`-like format, and using the existing WET software.

7.4 The Final Word

We have shown that excellent time resolved CCD photometry is not only possible, but, when run through software designed particularly for this purpose, can be as efficient and easy to interpret as classical PMT photometry. It is the ease of use and instant results that make this solution more elegant than what has been achieved in previous experiments on time resolved CCD photometry. Hopefully, other astronomers will see it the same way, and make sure that the solutions provided here will be used and developed further into a future standard for CCD photometry of variable stars.

Bibliography

[API97] American Precision Industries (API), Control Division: 1997, *IDRIVE Protocol Specification rev. 0.5*

[Abr90] Abrahamian H.V., Lipovetsky V.A., Mkaelian A.M., Stephanian J.A.: 1990, *Astrofizika*, **33**, 345

[And98] Andersen J.: 1998, in: *Optical Detectors for Astronomy*, eds. J.W. Beletic & P. Amico, Kluwer

[Ber80] Berger J., Fringant A.M.: 1980, *Astron. & Astrophys. Suppl. Ser.*, **39**, 39

[Bil97] Billères M., Fontaine G., Brassard P. *et al.*: 1997, *Astrophys. J. Letters*, **487**, L81

[Bil98] Billères M., Fontaine G., Brassard P. *et al.*: 1998, *Astrophys. J. Letters*, **494**, L75

[Bil00] Billères M., Fontaine G., Brassard P. *et al.*: 2000, *Astrophys. J.*, **530**, 441

[Bon95] Bond H.E.: 1995, *Baltic Astronomy*, **4**, 527

[Cha96] Charpinet S., Fontaine G., Brassard P., Dorman B.: 1996, *Astrophys. J.*, **471**, L106

[Cha97] Charpinet S., Fontaine G., Brassard P. *et al.*: 1997, *Astrophys. J.*, **483**, L123

[Cle93] Clemens J.C.: 1993, in: [Mei93a], 511

[Dor95] Dorman B., O'Connell R.W., Rood R.T.: 1995, *Astrophys. J.*, **442**, 105

[Dun85] Dunham E.W, Baron R.L., Elliot J.L. *et al.*: 1985, *Publ. Astron. Soc.-Pac.*, **97**, 1196

[Fon98] Fontaine G, Charpinet S., Brassard P. *et al.*: 1998, in: *New Eyes to See Inside the Sun and Stars*, eds, Deubner F.-L., Christensen-Dalsgaard J., Kurtz D., IAU Symp., **185**, 367

[Gic78] Giclas H.L., Burnham R., Thomas N.G.: 1978, *Lowell Obs. Bull.*, **8**, 51

[Gil93] Gilliand R.L., Brown T.M., Kjeldsen H. *et al.*: 1993, *Astron. J.*, **106**, 2241

[Gre86] Green R.F., Scmidt M., Liebert J.: 1986, *Astrophys. J. Suppl. Ser.*, **61**, 305

[Heb99] Heber U., Edelmann H., Lemke M. *et al.*: 1999, in: [Sol99], 551

[How92] Howell S.B.: 1992, in: *Astronomical CCD Observing and Reduction Techniques*, ed. Howell, S.B., ASP Conf. Series **23**, 105

[Kan00] Kanaan A., O'Donoghue D., Kleinman S.J. *et al.*: 2000, in: [Mei00], 387

[Kep93] Kepler S.O.: 1993, in: [Mei93a], 515

[Kil88] Kilkenny D., Heber U., Drilling J.S.: 1988, South African Astron. Obs. Circ., **12**, 1

[Kil97] Kilkenny D., Koen C., O'Donoghue D, Stobie R.S.: 1997, *Mon. Not.-Royal Astr. Soc.*, **285**, 640

[Kil98] Kilkenny D., O'Donoghue D., Koen C. *et al.*: 1998, *Mon. Not. Royal Astr. Soc.*, **296**, 329

[Kil99] Kilkenny D., Koen C., O'Donoghue D. *et al.*: 1999, *Mon. Not. Royal Astr. Soc.*, **303**, 525

[Kin83] King I.B.: 1983, *Publ. Astron. Soc. Pac.*, **95**, 163

[Kje92] Kjeldsen H. & Frandsen S.: 1992, *Publ. Astron. Soc. Pac.*, **104**, 413

[Kle95] Kleinmann S.J., Nather R.E., Phillips T.: 1995, in: [Mei95], 482

[Koe97] Koen C., Kilkenny D., O'Donoghue D. *et al.*: 1997, *Mon. Not. Royal Astr. Soc.*, **285**, 645

[Koe98a] Koen C., O'Donoghue D., Kilkenny D. *et al.*: 1998, *Mon. Not. Royal Astr. Soc.*, **296**, 317

[Koe98b] Koen C.: 1998, *Mon. Not. Royal Astr. Soc.*, **300**, 567

[Koe98c] Koen C., O'Donoghue D., Pollacco D.L., Nitta A.: 1998, *Mon. Not. Royal Astr. Soc.*, **300**, 1105

[Koe99a] Koen C., O'Donoghue D., Pollacco D.L, Charpinet S.: 1999 *Mon. Not.-Royal Astr. Soc.*, **305**, 28

[Koe99b] Koen C., O'Donoghue D., Kilkenny D., Stobie R.S., Saffer R.A.: 1999, *Mon. Not. Royal Astr. Soc.*, **306**, 213

[Lam00] Lamontagne R., Demers S., Wesemael F., Fontaine G., Irwin M.J.: 2000, *Astron. J.*, **119**, 241

[McL89] McLean I.S.: 1989, *Electronic and Computer Aided Astronomy*, Ellis Horwood Ltd.

[Mei93a] Meištas E.G. & Solheim J.-E.: 1993, *Proceedings of the 2nd WET Workshop*, *Baltic Astronomy*, **2**, Nos. 3–4

[Mei93b] Meištas E.G. & Solheim J.-E.: 1993, in: *White Dwarfs: Advances in Observations and Theory*, ed. M.A. Barstow, Kluwer, 549

[Mei93c] Meištas E.G.: 1993, in: [MEI93a], 498

[Mei95] Meištas E.G. & Solheim J.-E.: 1995, *Proceedings of the 3rd WET Workshop*, *Baltic Astronomy*, **4**, Nos. 2–4

[Mei00] Meištas E.G. & Vauclair G. (eds): 2000, *Proceedings of the 5th WET Workshop*, *Baltic Astronomy*, **9**, Nos. 1–3

[Moe90] Moehler S., Richtler T., de Boer K.S., Dettmar R.J., Heber U.: 1990 *Astron. & Astrophys. Suppl. Ser.*, **86**, 53

[Myr79] Myrabø H.K.: 1979, *Astrop. & Space Sci.*, **60**, 367

[Myr80] Myrabø H.K.: 1980, *Astron. & Astrophys.*, **84**, 297

[Nat90] Nather R.E., Winget D.E., Clemens J.C.: 1990, *Astrophys. J.*, **361**, 309

[New91] Newberry M.V.: 1991, *Publ. Astron. Soc. Pac.*, **103**, 122

[Nay98] Naylor T.: 1998, *Mon. Not. Royal Astr. Soc.*, **296**, 339

[ODo95] O'Donoghue D.: 1995, *Baltic Astronomy*, **4**, 519

[ODo97] O'Donoghue D., Lynas-Gray A.E, Kilkenny D. *et al.*: 1997, *Mon. Not.-Royal Astr. Soc.*, **285**, 657

[ODo98a] O'Donoghue D., Koen C., Solheim J.-E. *et al.*: 1998, *Mon. Not. Royal Astr. Soc.*, **296**, 296

[ODo98b] O'Donoghue D., Koen C., Lynas-Gray A.E. *et al.*: 1998, *Mon. Not. Royal Astr. Soc.*, **296**, 306

[ODo99] O'Donoghue D. *et al.*: 1999, in: [SOL99], 149

[ODo00] O'Donoghue D., Kanaan A., Kleinmann S.J., Krzesinski J., Pritchet C.: 2000, in: [MEI00], 375

[OEs96] Østensen R., Refsdal S., Stabell R., Teuber J. *et al.*: 1996, Å**308**, 49

[OEs00a] Østensen R., Solheim J.-E.: 2000, in: [MEI00], 411

[OEs00b] Østensen R., Solheim J.-E., Silvotti R., Heber U., Dreizler S., Edelmann H.: 2000, submitted to *Astronomy & Astrophysics*

[Pic00] Piccioni A., Bartolini C., Guarnieri A. *et al.*: 2000, *Astronomy & Astrophysics*, **354**, L13

[Pre92] Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P.: 1992, Numerical Recipies in C, 2^{nd} ed., Cambridge

[Ree00] Reed M.D. *et al.*: 2000, in: [MEI00], 179

[Rob87] Roberts D.H., Lehár J., Dreher J.W.: 1987, *The Astronomical Journal*, **93**, 968

[Roq00] Roques S., Schwarzenberg-Czerny A., Doles N.: 2000, in: [MEI00], 463

[Saf94] Saffer R.A., Bergeron P., Koester D., Liebert J., 1994, *Astrophys. J.*, **432**, 351

[Sil00a] Silvotti R., Gonzales-Perez J.M., Solheim J.-E., Heber U., Dreizler S., Edelmann H.: 2000, in: [MEI00], 205

[Sil00b] Silvotti R. *et al.*: 2000, *Astronomy & Astrophysics*, **359**, 1068

[Sol99] Solheim J.-E. & Meištas E.G. (eds): 1999, *Proceedings of the 11*$^{\text{th}}$ *European Workshop on White Dwarfs*, ASP Conf. Series, **169**

[Sto97] Stobie R.S., Kawaler S.D., Kilkenny D. *et al.*: 1997,

[Sto87] Stover R.J. & Allen S.L.: 1987, *Mon. Not. Royal Astr. Soc.*, **285**, 651

[Tek91] Tektronix Inc.: 1991, *TK1024 CCD Imager*, Tektronix specification document

[Teu93] Teuber J.: 1993, *Digital Image Processing*, Prentice-Hall

[Tho96] Thomassen T.: 1996, *Presisjons fotometri ved Skibotn Observatorium*, Master thesis, University of Tromsø (in Norwegian)

[War88] Warner B.: 1988, *High Speed Astronomical Photometry*, Cambridge

[Weg86] Wegner G., McMahon R.G.: 1986, *Astron. J.*, **91**, 139

[Wel81] Wells D.C., Greisen E.W., Harten R.H.: 1981, *Astron. & Astrophys.-Suppl. Ser.*, **44**, 363

[Wis91] Wisotzki L., Wamsteker W., Reimers D.: 1991, *Astron. & Astrophys.*, **247**, L17

[You67] Young A.T.: 1967, *Astron. J.*, **72**, 747