Ongoing Software Development Projects under Linux/QNX

1. CCD Image acquistion software under Linux

   i) Gtk version(2.0) ii) IDL version (6.3)

   -V.Arumugam, A.V.Ananth

2. Software development under QNX REAL-TIME O.S. Although our efforts are for CONTROL applications specifically for HAGAR, as an initial step we have tried CCD image acquistion software development in QNX to familiarise with various features like Photon Application Builder(PhAB), Resource Manager, Process Manager etc.

   -V.Arumugam, A.V.Ananth

3. Distributed Telescope control system for 30" telescope at VainuBappu Observatory.The approach we have planned is similar to that of 2 Mtr. Telescope.The advantage of this is scheme is its enoromous flexibility in implementation in the sense ,that different sub-systems could be modified/implemented at different points of time without substantially modify software related to other sub-systems.Also back-ends can also communicate with the telescope and other sub-systems if required to tap information.

   -V.Arumugam, A.V.Ananth, KavithaPathak, Faseehana, Anbazhagan

CCD Image acquisition software under Linux

i) Gtk version (2.0)

ii) IDL version (6.3)

MODEL: It is a Client/Server Design where the Server can be local or at remote site and client accesses the server on LAN/WAN.

Applicability: This software intended for 2Kx2K TEK chip and 2Kx4K Marconi chip. Can be used with a little modification for any other sensor in future if appropriate Hardware modifications are made for the existing controller.

Drawbacks: As linux and gtk/genome are freely available software and are continuously being modified their version keep changing and hence maintenance of camera software is a hectic task.

Moreover Gtk coding is complex and difficult to maintain.

IDL : Interactive Data Language developed by Research System Inc(www.rsinc.com).

Version used: 6.3

Advantages: very simple function calls, API and commands. Built-in image analysis and processing functions and produces versatile display.

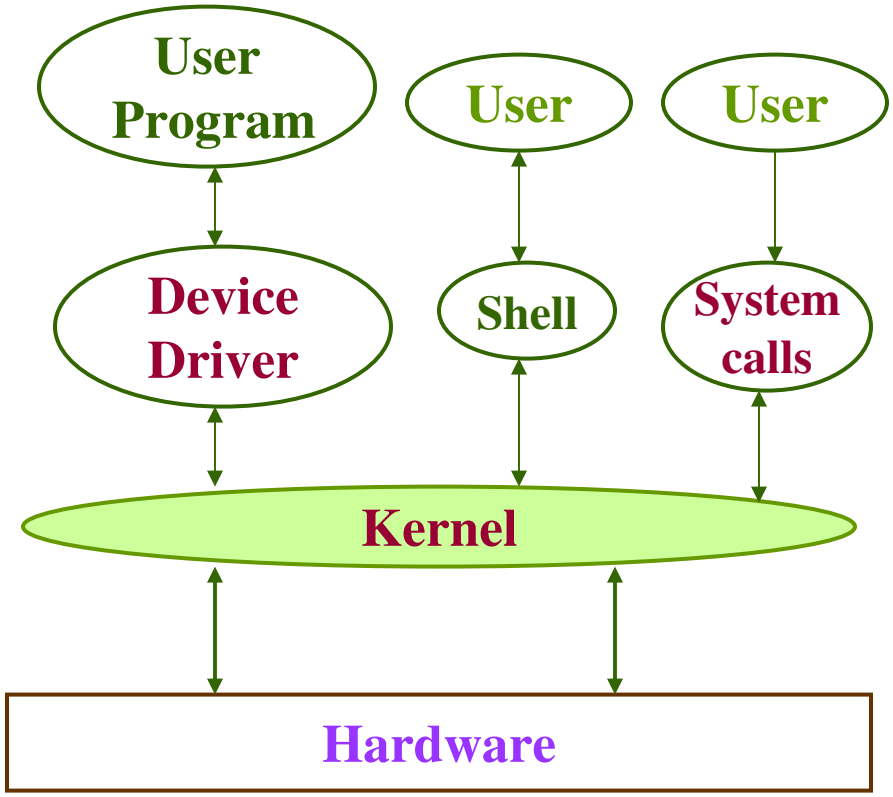
Drawback: It is costly, because every license is chargeable. Compatibility with C/C++ is not straight-forward. IDL programs can not call C routines directly.

C programs are converted into C-objects, IDL program call these C-objects to perform intended task.

Memory consumed by the software is little bit high.

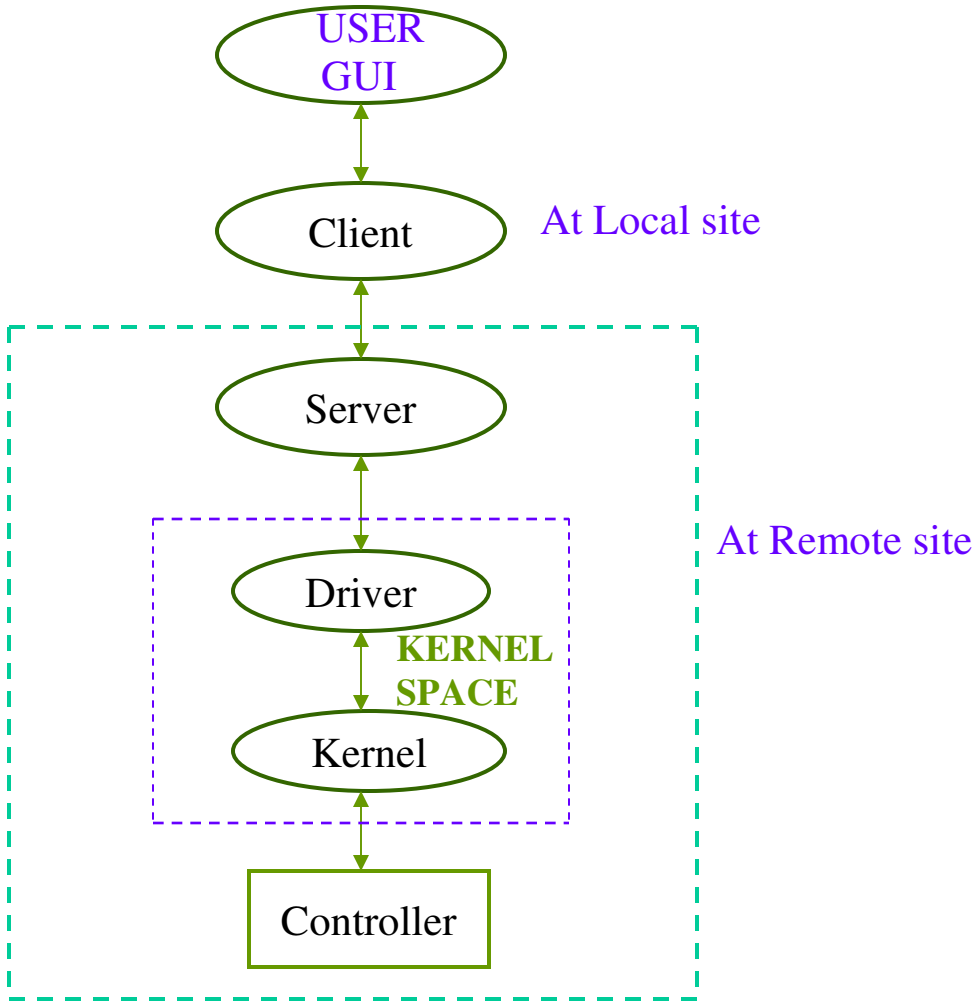
Good resolution monitor with large Video RAM is needed.

General Process Communication Concept in LINUX

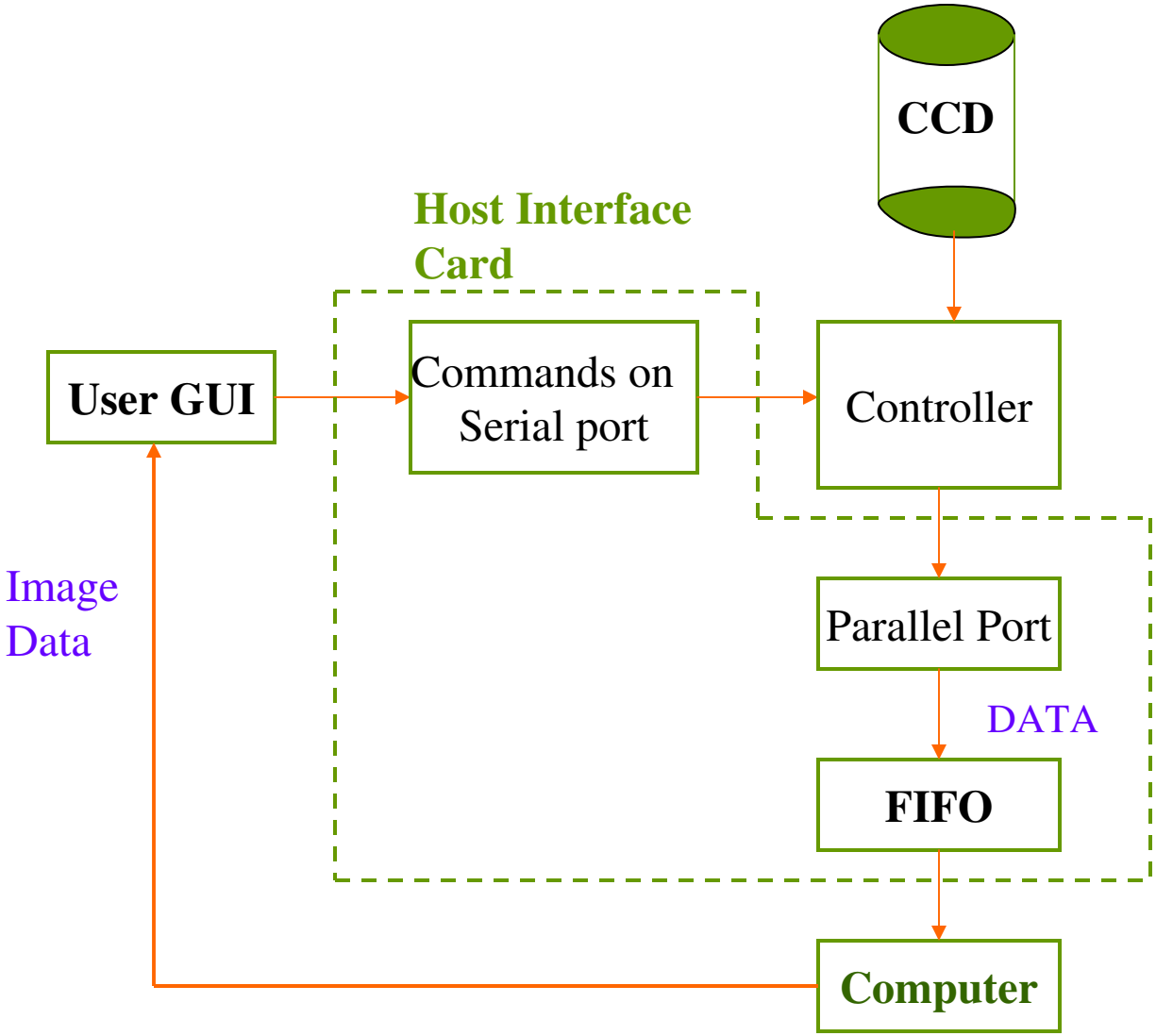


User Programs invokes IOCTL calls to talk to the Device driver which inturn talk to the kernel to communicate to the Hardware

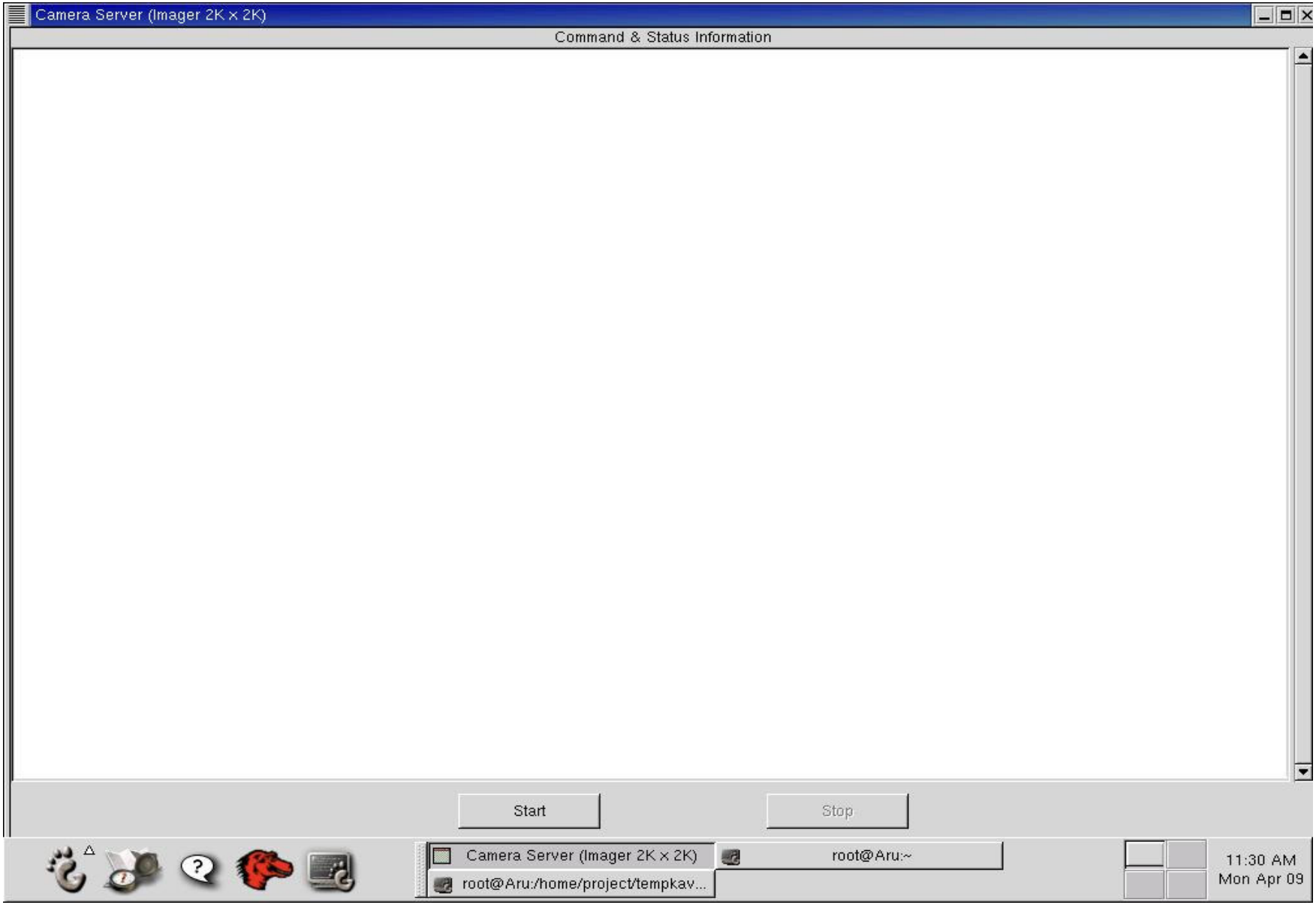
Scheme for CCD Image Acquisition Software under Linux



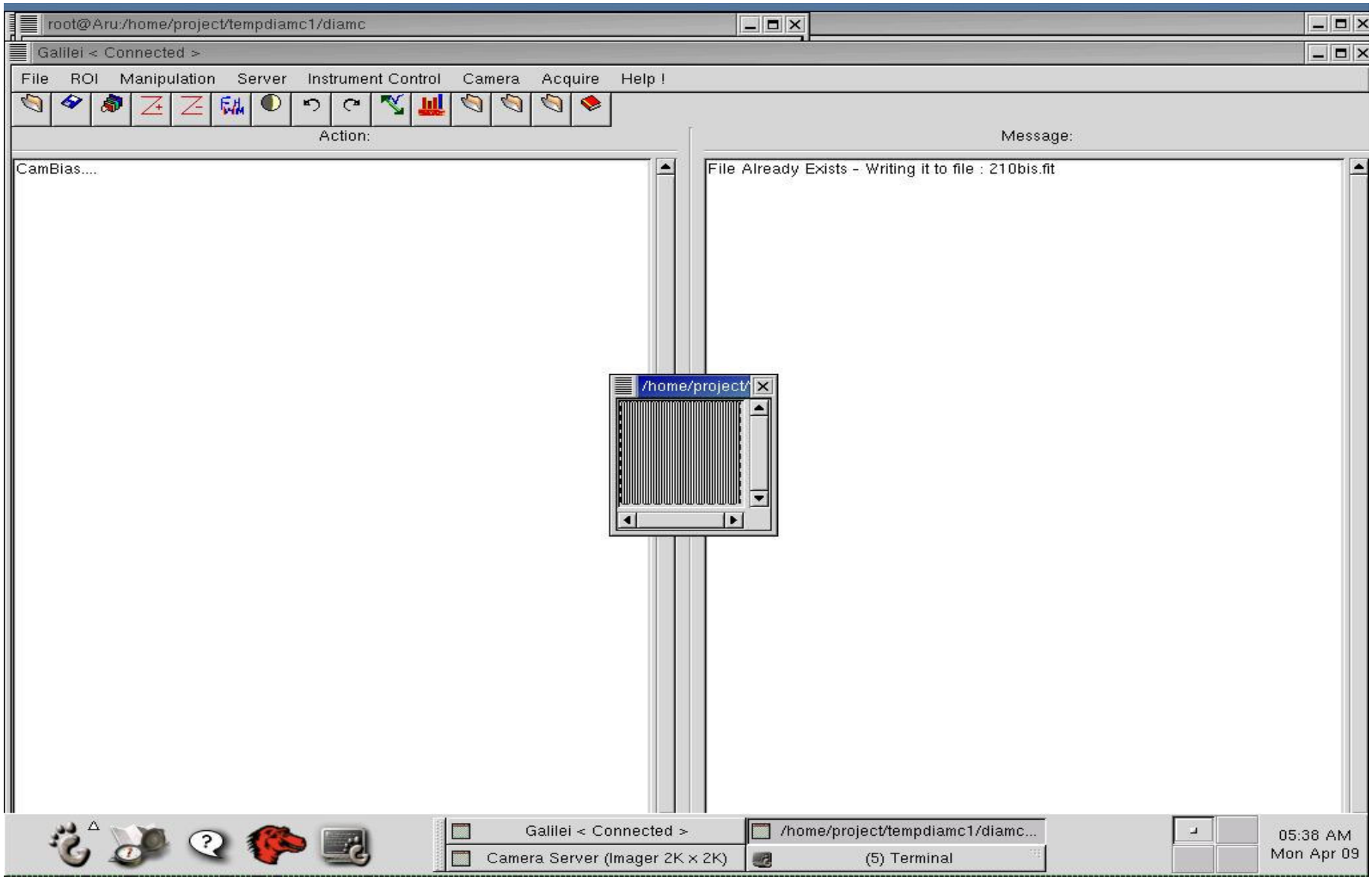
Hardware arrangement for CCD Image Acquisition



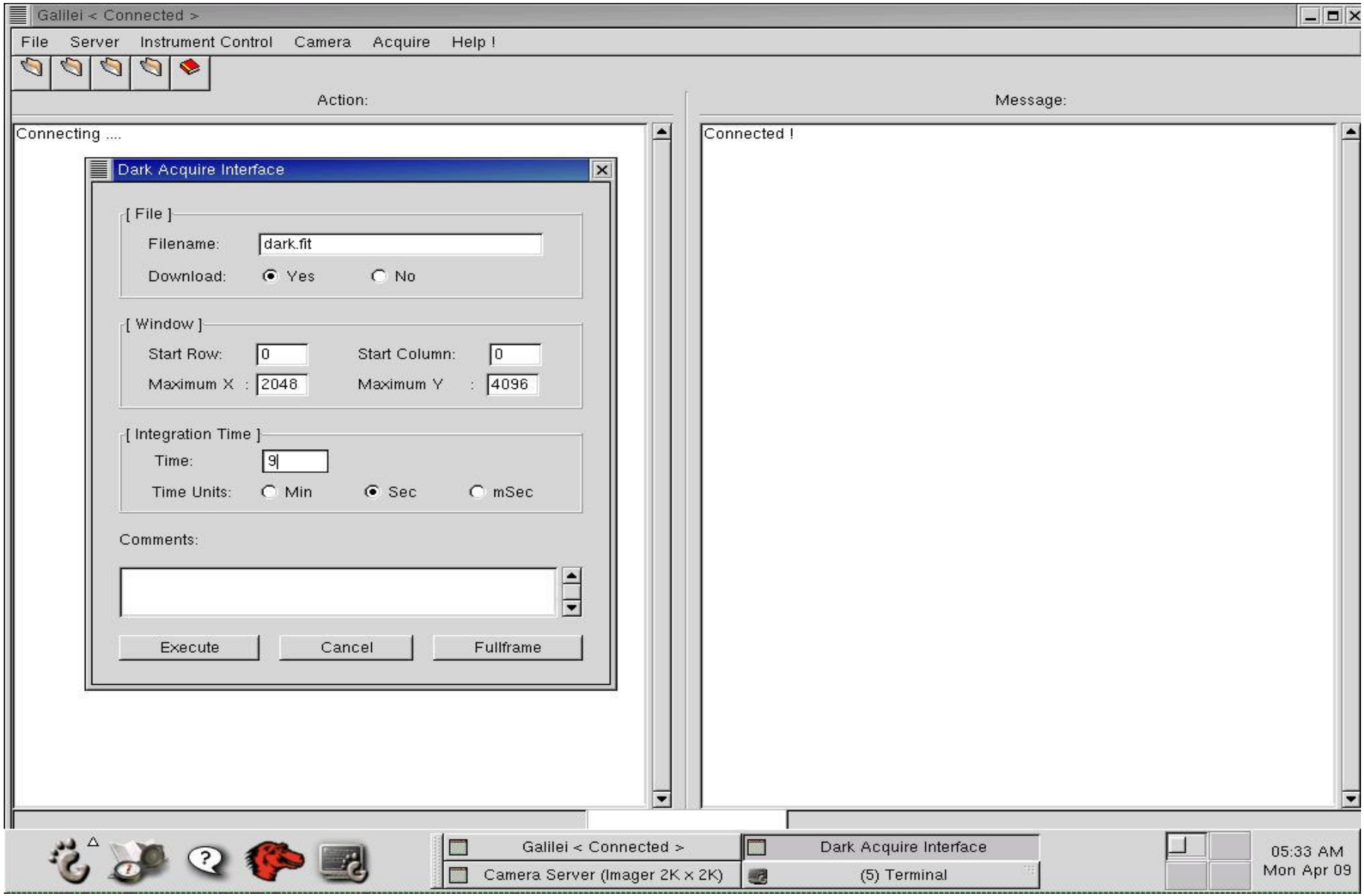
Output of CCD Image Acquisition Server – Gtk Version



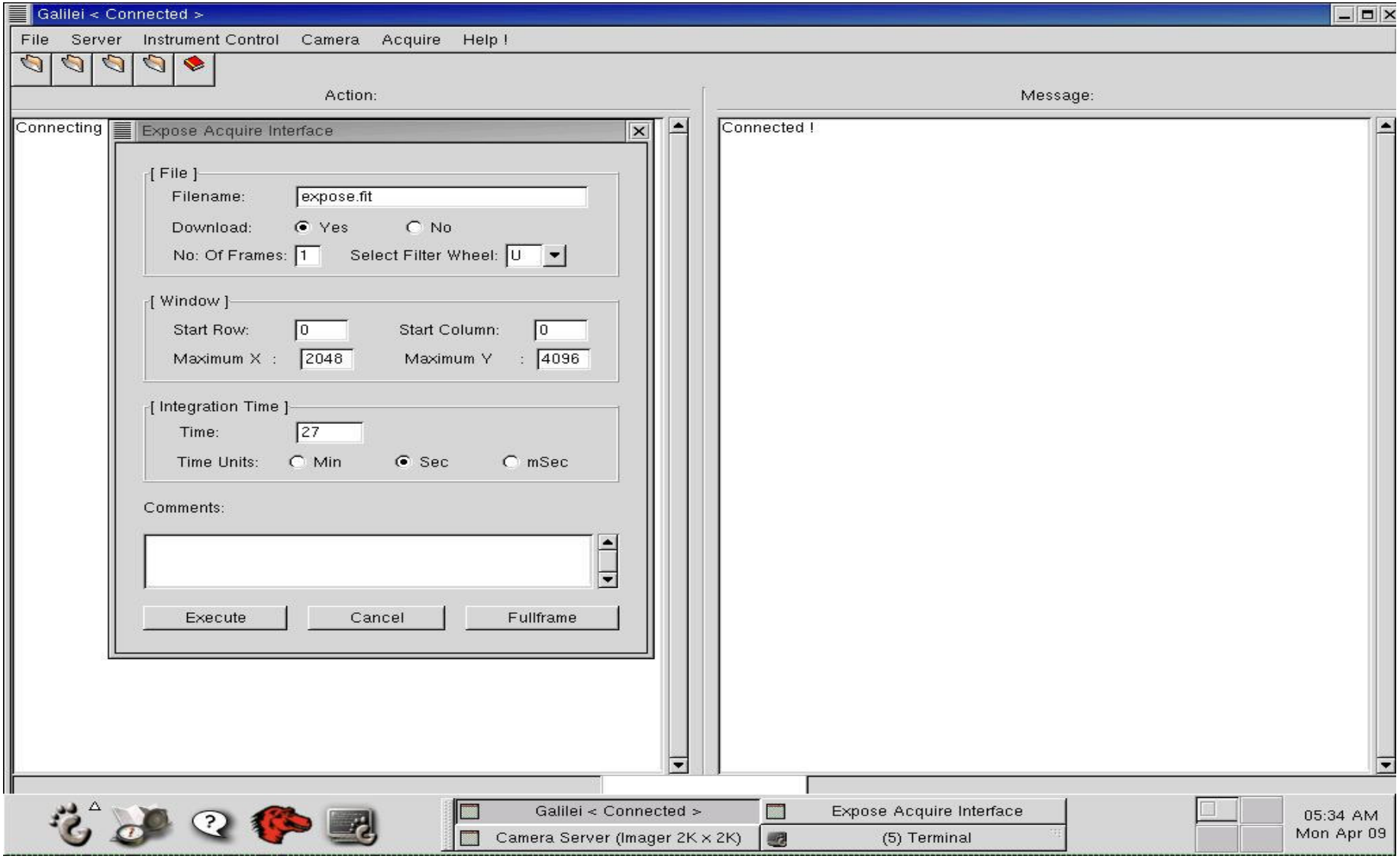
Client which acquired a bias frame of size 90x90



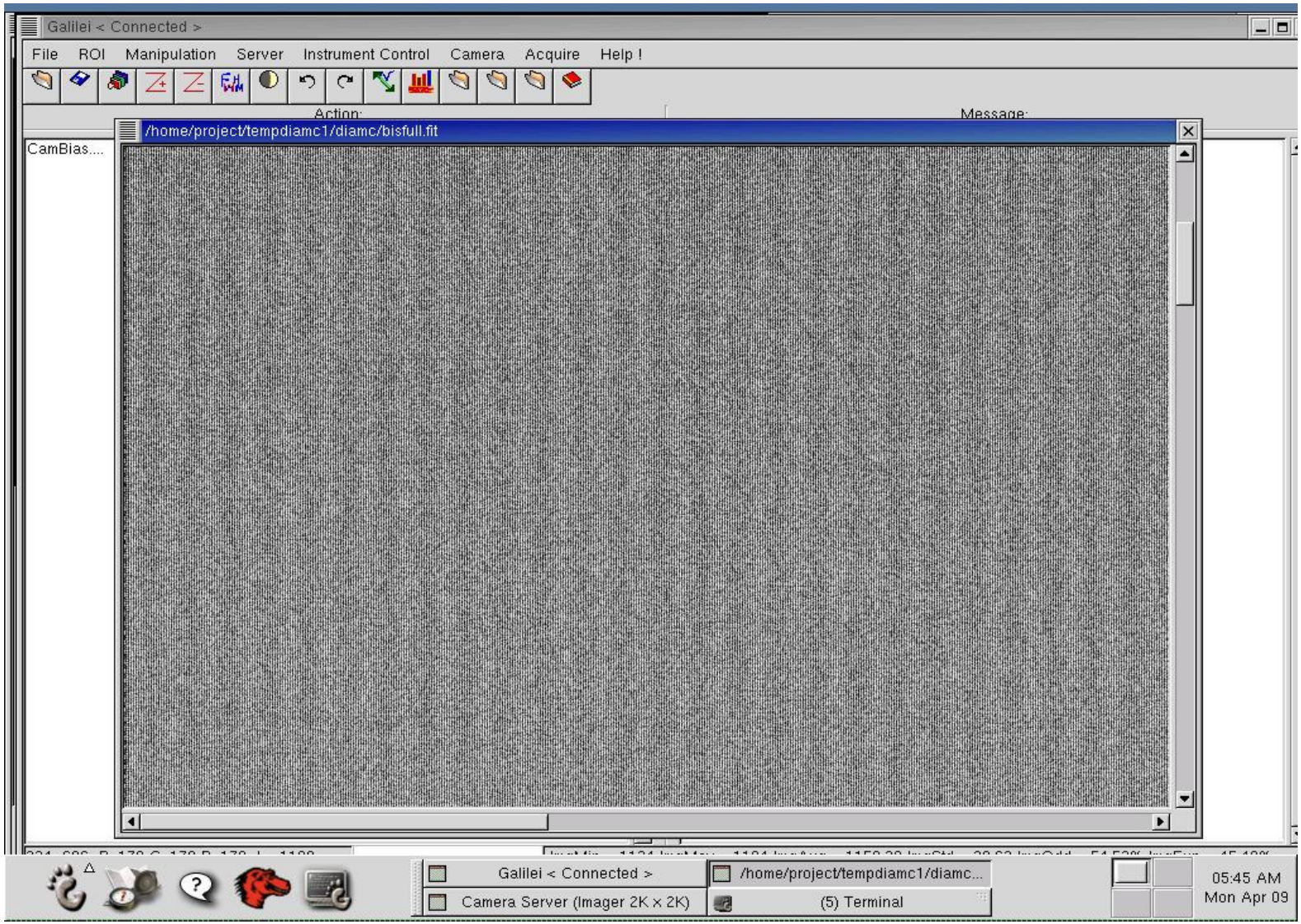
A dark window ready to acquire a frame of size 2Kx4K



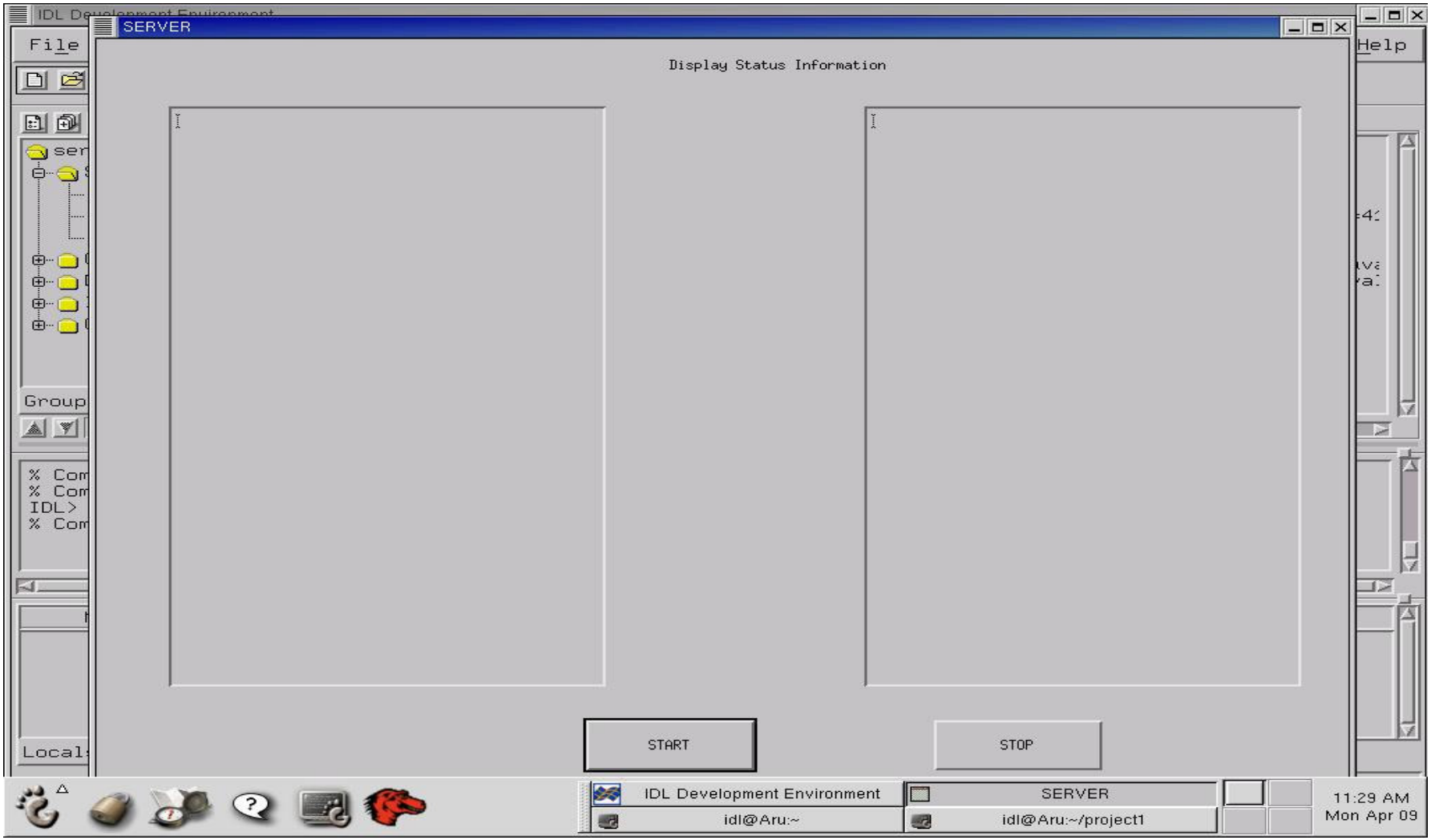
A window ready to acquire a frame of size 2Kx4K



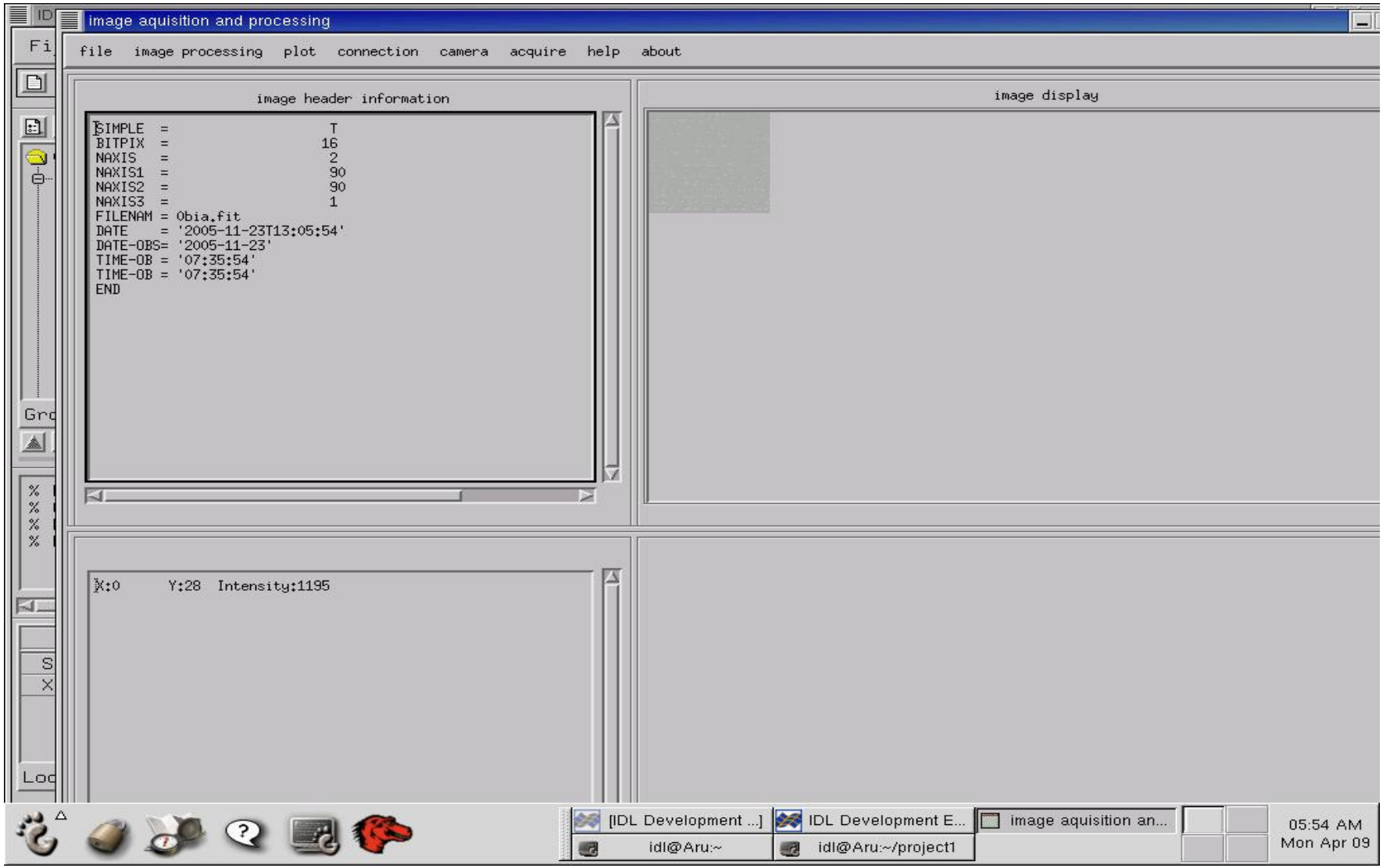
A bias frame of size 2Kx4K (GTK version)



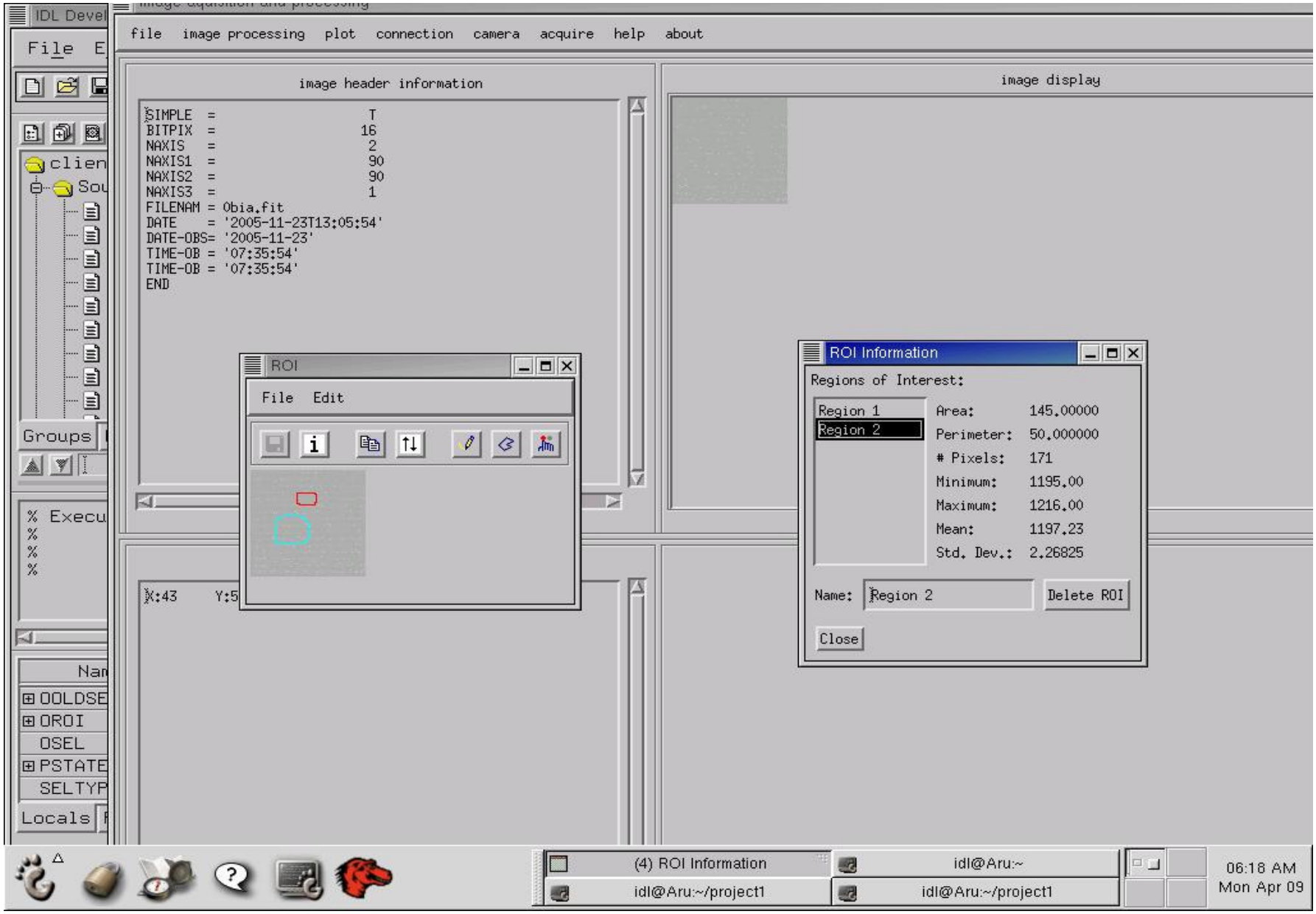
Output of IDL version of server



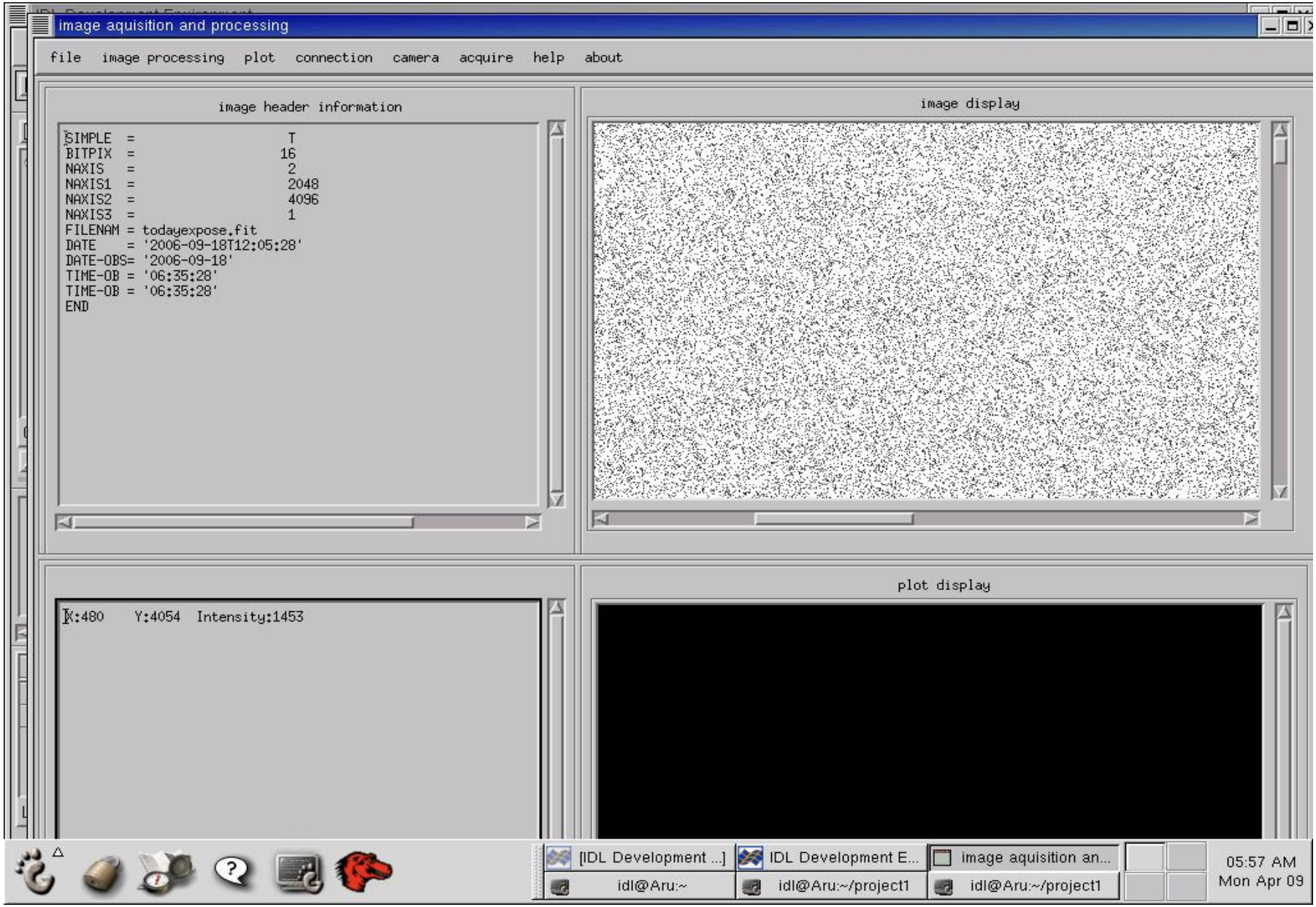
Client which acquired a bias frame of size 90x90 (IDL)



A sample ROI facility for a small frame 90x90 (IDL version)



A bias full frame of size 2Kx4K (IDL version)



Performance

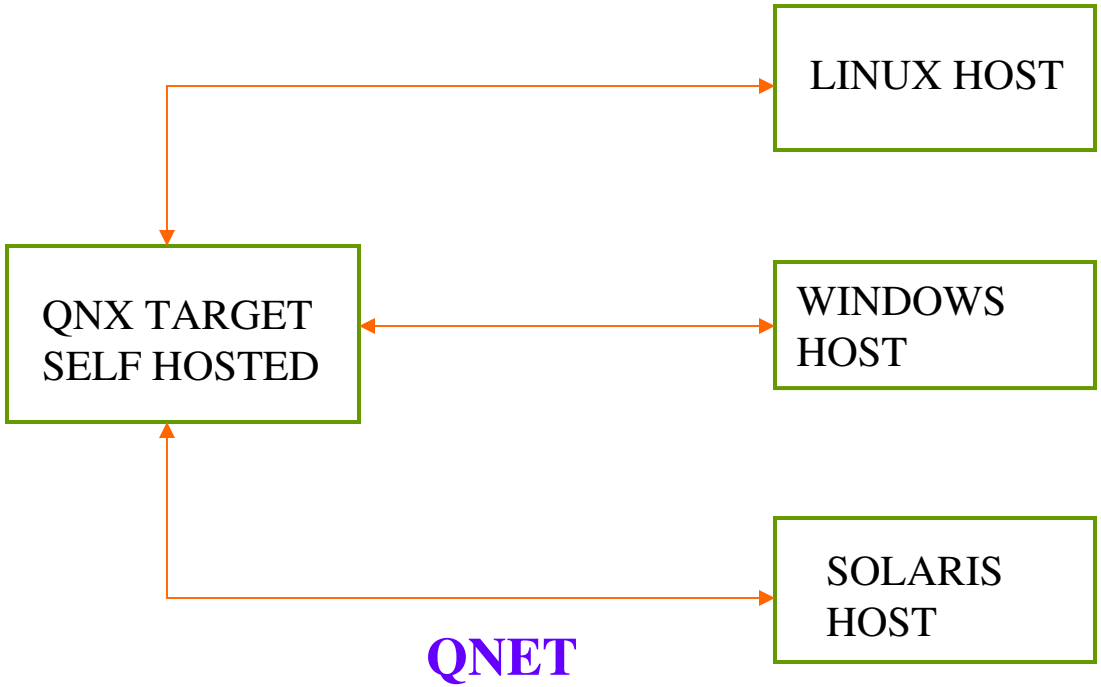
The data transfer rate is 2 minutes 20 seconds for 2Kx4K image with 2 readouts and 4 minutes 23 seconds for single readouts

The data transfer rate is 2 minutes 25 seconds for 2Kx2K image for a single readout since, there is no multiple readout capability.

Status

The system is currently undergoing tests for multi-read out operations.

QNX Real-Time Operating System primarily for embedded applications but here we are using for desktop. It compares in performance with other Real-Time OS like VxWorks, PSoS, RT-Linux, Windows-CE



Linux,Window,Solaris hosts are purely for development environment.

PHOTON MICRO GUI

i) IDE ii) Photon Application Builder(PhAB)

Why QNX?

Because QNX is POSIX compliant and original HAGAR software was under Linux, hence porting is relatively simple.

Advantages of QNX:

1. **High Availability: Small or large, QNX installed systems have high availability because of fault tolerant features. Can be accommodated in systems with very limited resources.**
2. **Message Passing – Message Passing virtual software bus that lets you dynamically plug in, or plug out, any component on the fly.**
3. **POSIX Compliance – It enables the creation of reusable software assets if coded in UNIX/LINUX and improve productivity, and accelerate product delivery.**
4. **Transparent Distributed Processing – It provides a framework for the dynamic interconnection of hardware and software resource located on remote nodes, using standard messages. Processes running on a single CPU will continue to communicate with each other even if they are subsequently distributed among multiple CPUs. With this unique capability creation of robust and fault-tolerant systems that offer on-demand access to resources on multiple CPUs is possible.**

5. **Symmetric multiprocessing** – It is the only commercial RTOS to support true SMP, providing a huge performance boost to compute-intensive systems involving Real-Time Applications.

6. **Advanced Graphics – Photon MicroGUI**

7. **Critical Process Monitoring(CPM)** – QNX Neutrino's modular,microkernel architecture enables the isolation of faults right down to the driver level. Together with CPM,"smart watchdog" technology that helps system recover from faults automatically. This approach enables the development of truly **self-healing** systems.

8. **Networking Technologies** – It provides a comprehensive suite of natively supported networking protocols based on industry standard implementations.

9. **Resource Manager** – It provides a simple mechanism to write drivers for custom hardwired boards.

10. **Java Environment – Has J2ME support.**

Processors supported by QNX

i) (little endian) X86

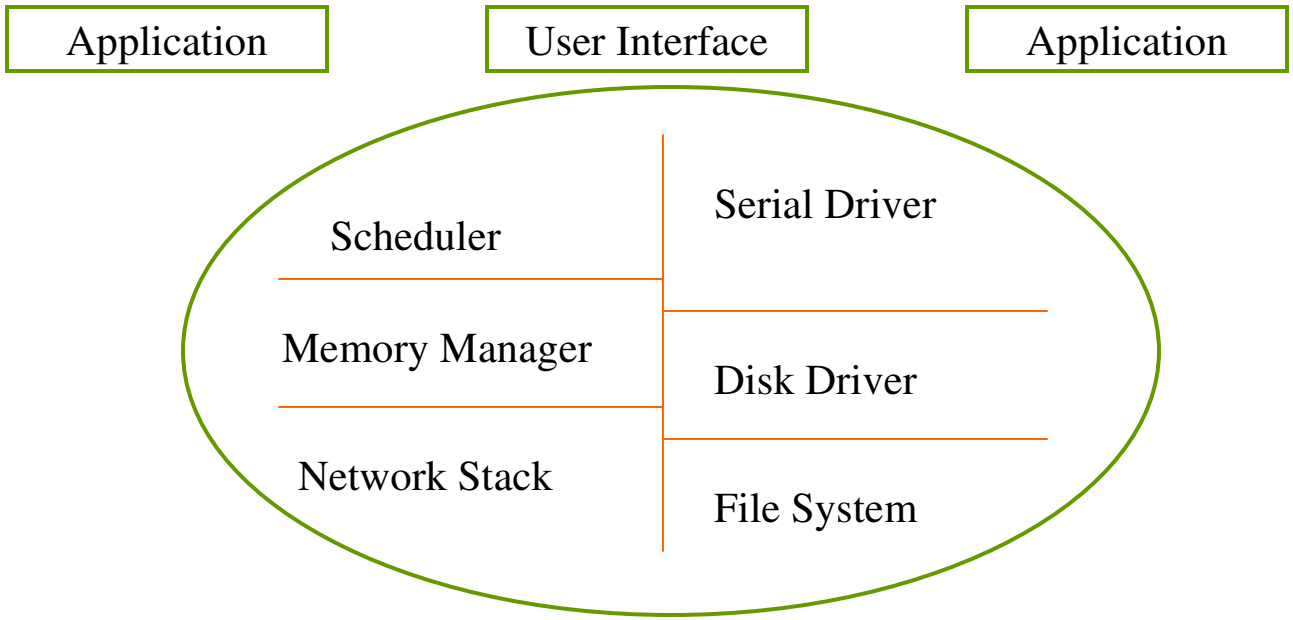
ii) ARM

iii) MIPS

iv) PPC(big endian)

v) SH(little endian)

Traditional Monolithic kernel

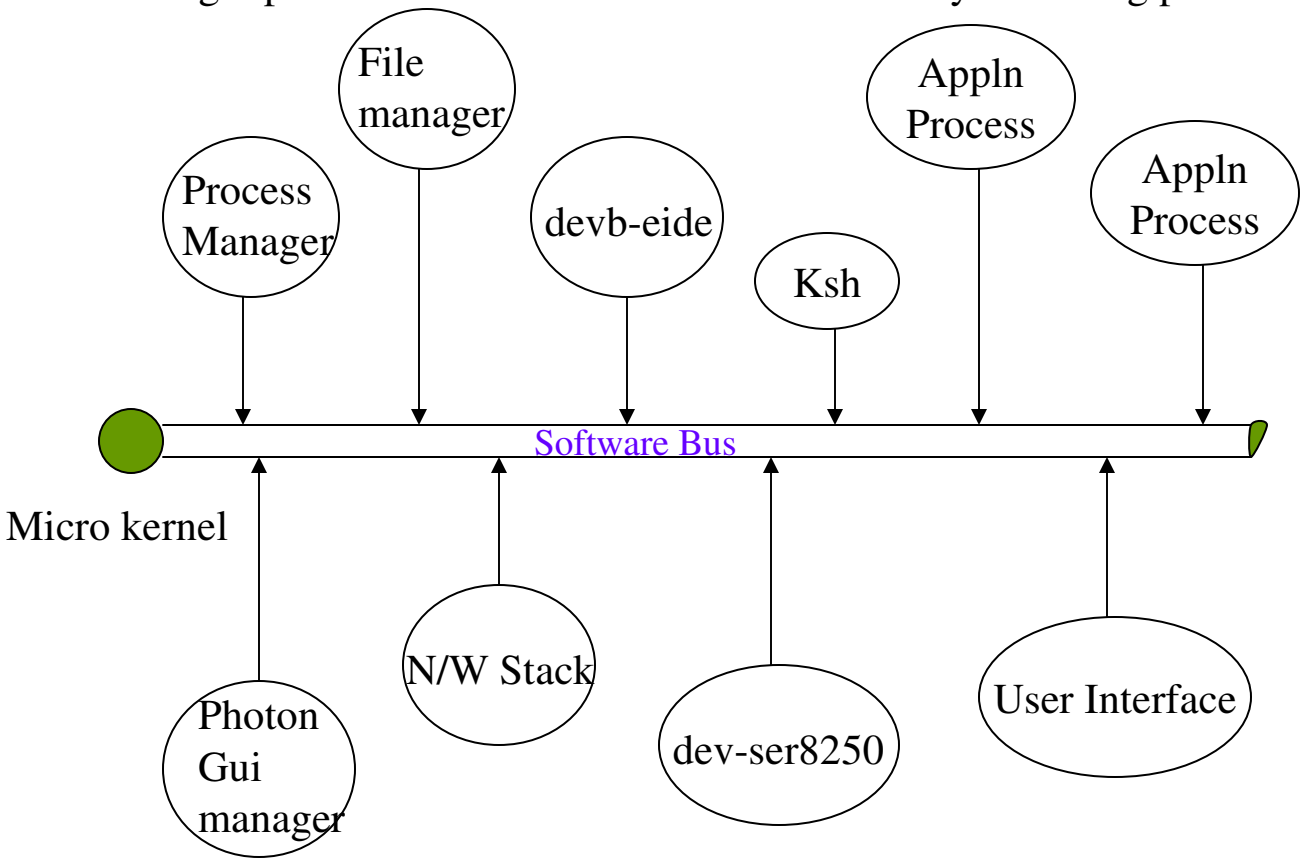


Problems with Monolithic Kernels

The kernel contains the OS kernel functionality and all the drivers. so driver development is complex and debugging can be painful and any failure with the device driver would adversely affect the functionality of the whole system.

Applications are processes in protected memory space, so the kernel is protected from applications and applications are protected from each other.

The ProcessManager places information about all the currently executing processes at /proc filesystem.



Interaction between components in QNX RTOS

The Micro-kernel manages a group of cooperating processes. QNX neutrino acts as a kind of software bus that lets us dynamically plug in/out OS modules whenever they are needed. This is a very important aspect for embedded systems.

Processes are separate from kernel so if something goes wrong it will never affect kernel.

MICROKERNEL – SERVICE

Microkernel is very small it is dedicated to only a few fundamental services.

Thread services via POSIX

Signal services via POSIX

Message-passing services

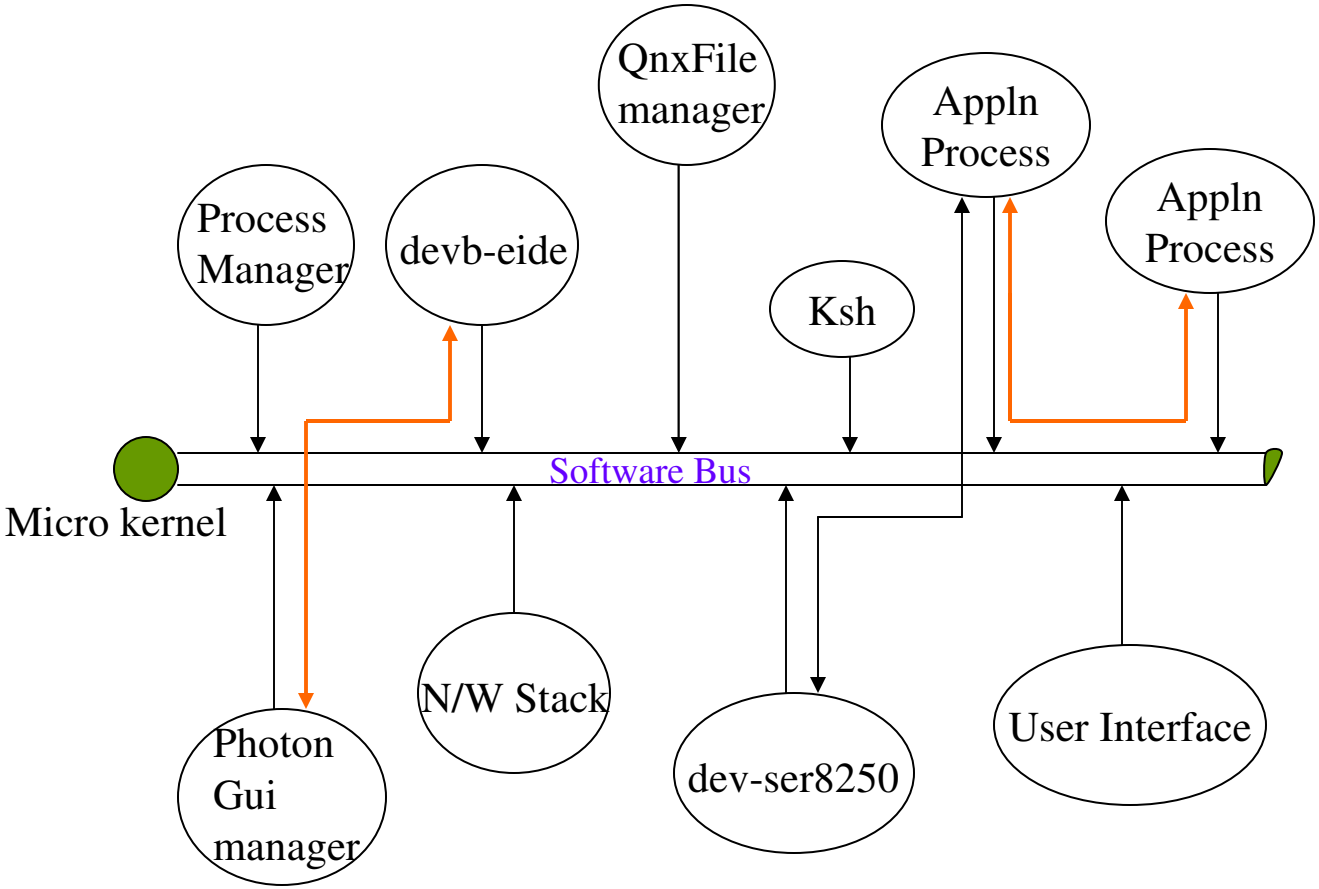
Synchronization services

Scheduling services – Microkernel schedules threads for execution using the various POSIX real-time scheduling algorithms.

Timer services – provides rich set of POSIX timer services.

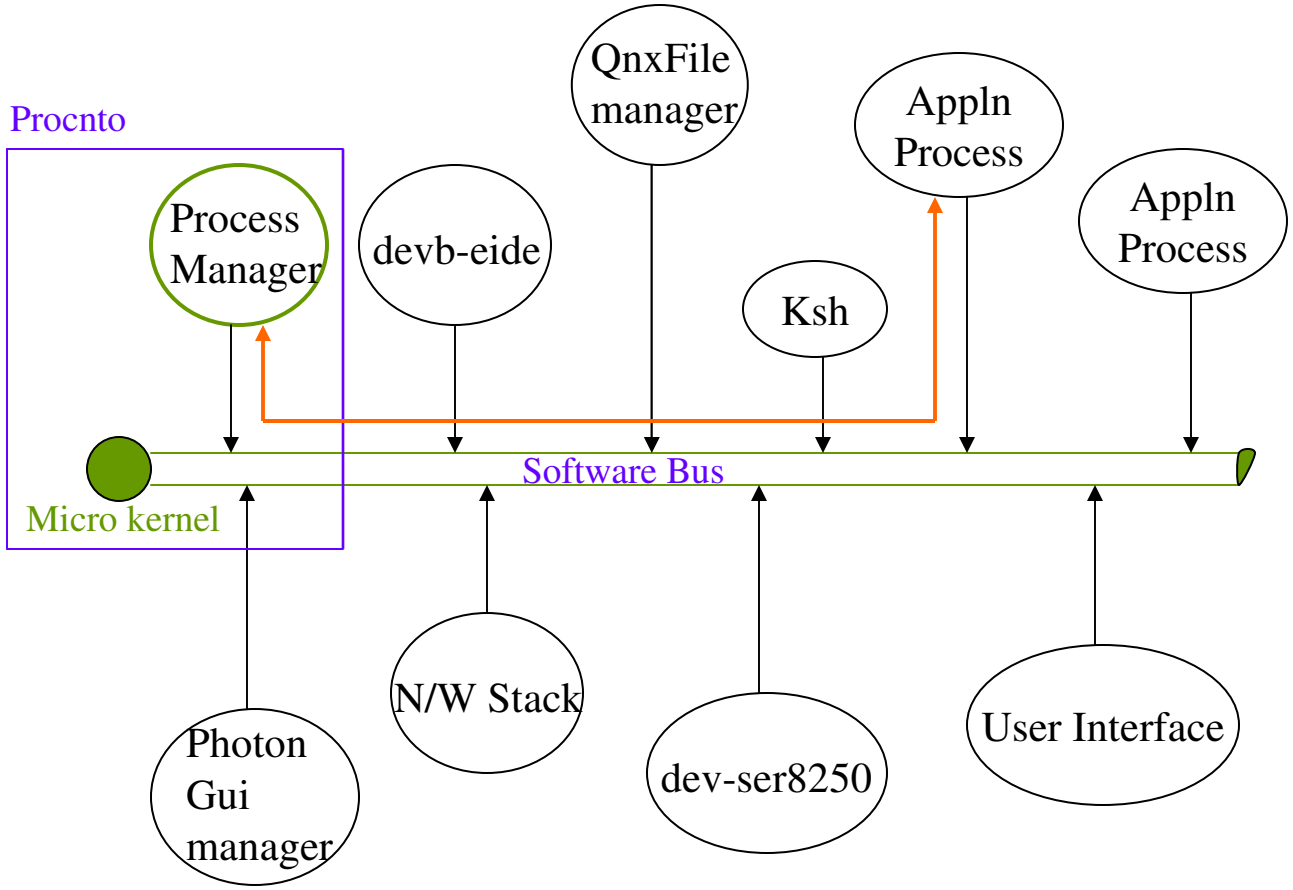
Process Management service – The Micro-kernel and Process Manager together form a unit called **Procnto.**

Processes communicate with each other



Message Passing – It forms a virtual "software bus" that let you dynamically plug in, or plug out, any component on the fly. Enabled by its message-passing design, QNX Neutrino implements only the most fundamental services in the OS kernel, such as signals, timers, and scheduling.

Communication with Process Manager

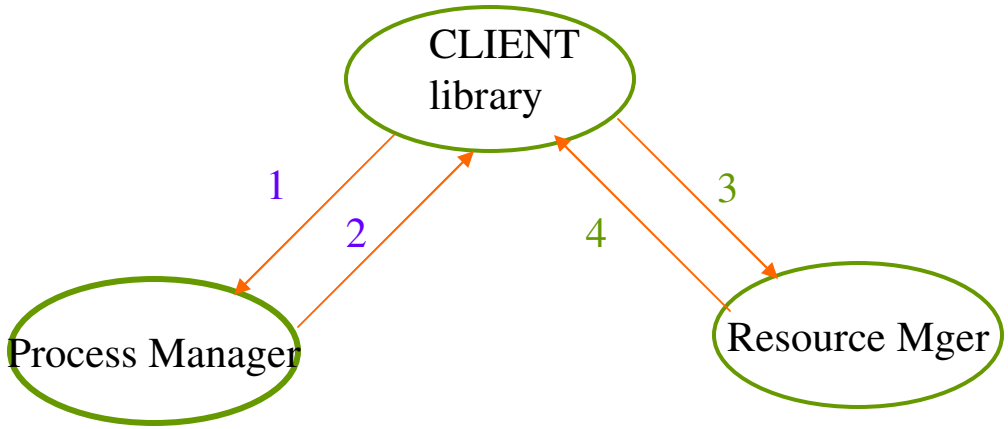


Processes communicates with ProcessManager using Message-passing IPC

Process Manager provides

- Process creation and termination e.g: spawn/exec/fork.
- Pathname management
- Memory protection, address space management
- Packaging of groups of threads together into processes

RESOURCE MANAGER – A user-level server program that accepts messages from other programs and , optionally, communicate with hardware, it is a process that registers a pathname prefix in the pathname space(e.g /dev/ser1),other process can communicate to it.



1. Client library(open()) sends a “query” message

2. Process Manager replies with who is responsible

3. Client’s library establishes a connection to the specified resource manager and sends an open message.

4. Resource manager responds with status(pass / fail)

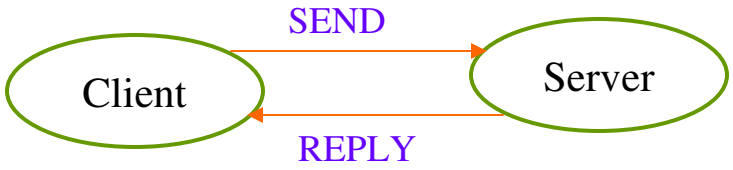
Kernel calls are pre-emptible.

Benefits:

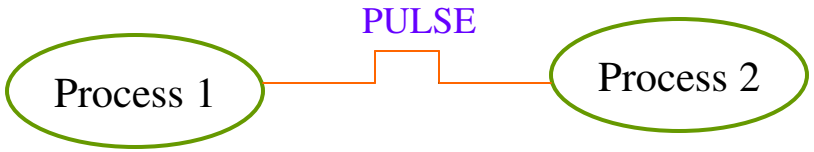
1. Responds to external events faster
2. Shorter interrupt latency(order of micro seconds)
3. Low Scheduling latency

IPC provided by the kernel

Messages – exchanging information between processes.



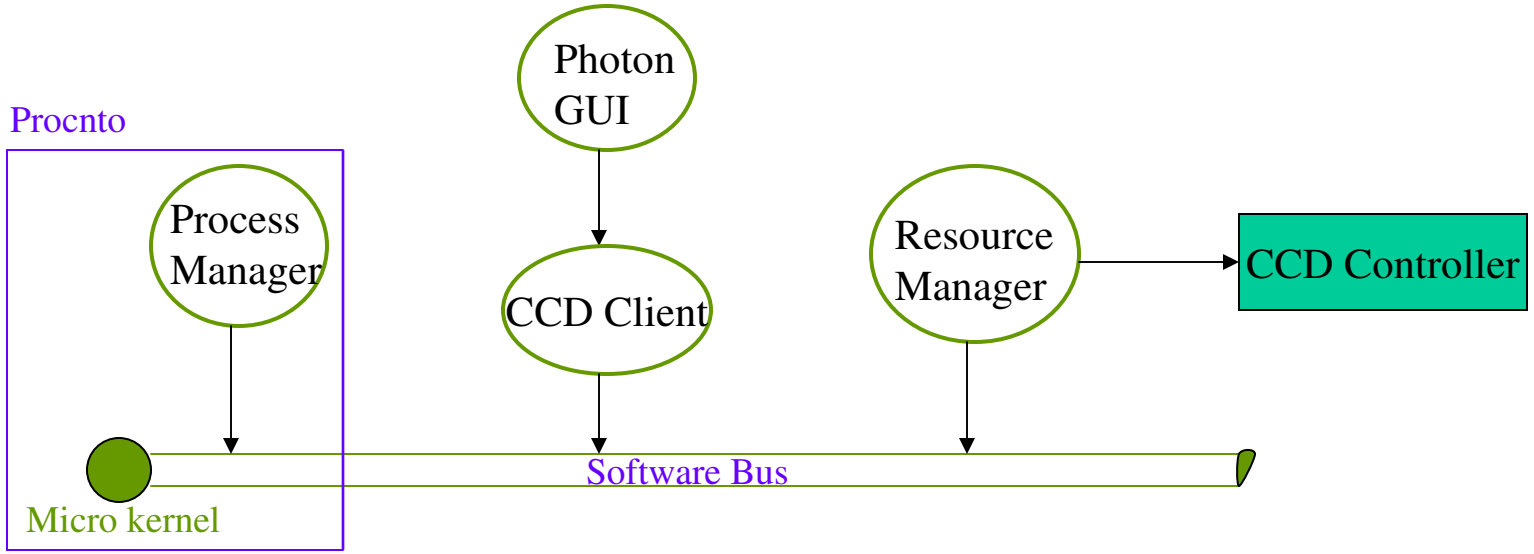
Pulses – used for event notification but asynchronous



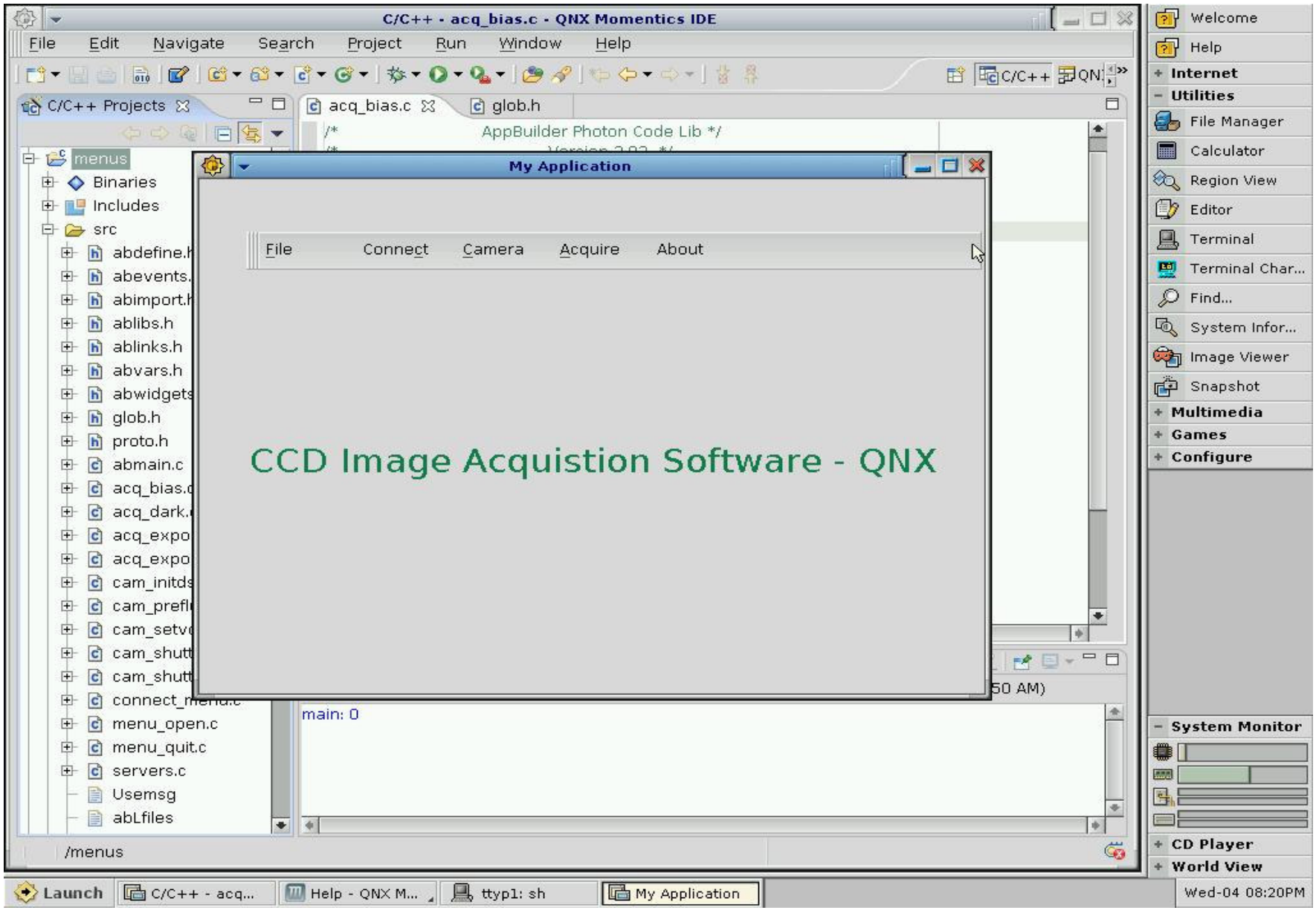
Signal – Interrupts another processes

Other IPC includes Message Queues,Shared Memory, Pipes, FIFOS

High level design of QNX version of CCD software

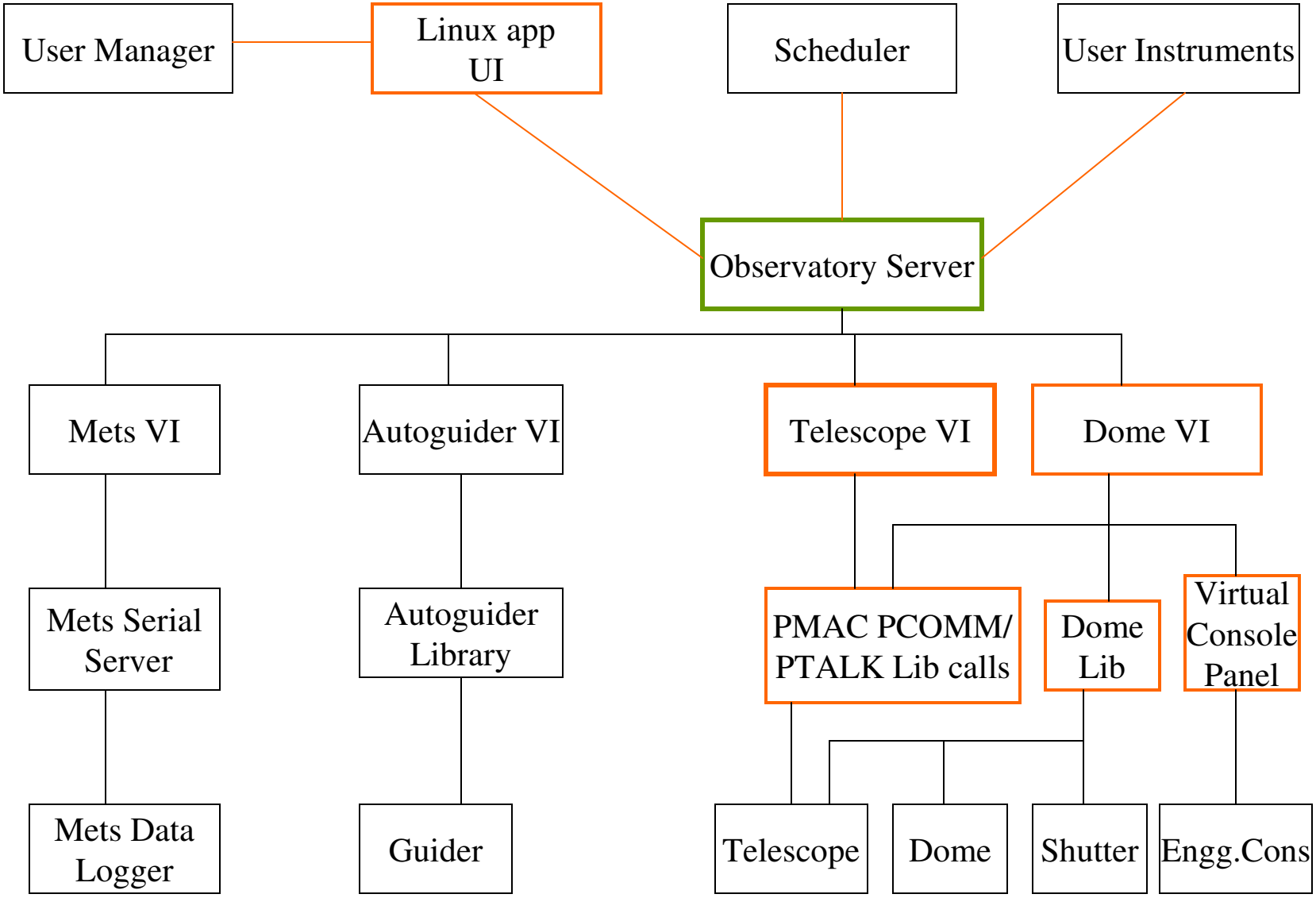


GUI for QNX version of CCD Software



Status of QNX version of CCD software:

At present the CCD Imaging software acquires the image data but needs to be analyzed on a LINUX system.



Software/Hardware Architecture for 30” Telescope, VBO, Kavalur

Operational Modes: Each Sub-system works in two modes viz.
Independent/Local Mode And **Remote Operation mode**

Independent Mode: The Telescope and dome are controlled by the system which houses the telescope control hardware(Turbo PMAC). This mode is also called Local Mode.

The operations are initiated through a GUI called virtual control panel(VCP);

The user inputs are accepted in the form of coordinate positions(RA and DEC). This mode is useful for independent testing of Telescope and Dome operations.

Remote Mode: An Observatory server gets user inputs and triggers the telescope and dome operations, it acts as a repository of all status information from various subsystems, which are periodically updated.

Features of Observatory server:

1. It works on TCP/IP sockets
2. Concurrent server
3. Multi-threaded program
4. Uses Message passing paradigm
5. Variable ASCII message size with upper limit of 512 bytes

6. It is suitable for limited volume data and not for bulk data like image data.

7. Can communicate with clients working under Window NT, Windows XP, Linux, Solaris, QNX.

Telescope Control Server:

The telescope control software runs as a multithreaded application. The application gets commandeered from the observatory server, user interface is through linux UI and, gets input from METs and the Autoguider systems. The Telescope control server houses the PMAC board(Programmable motion control boards from DeltaTau).

Dome Control Server:

The Hardware used for control of dome is a siemens inverter to drive the AC motors, and the scheme planned is very similar to 2M Dome.

Tools that are available for programming PMAC are

1. **PEWIN32** – testing the PMAC interactively from command prompt
2. **PCOMM32** – programming using VC++ under Window NT.

Status

The Design of Observatory server is in progress, Telescope control and Dome control software in local/autonomous mode is developed and ready for testing.