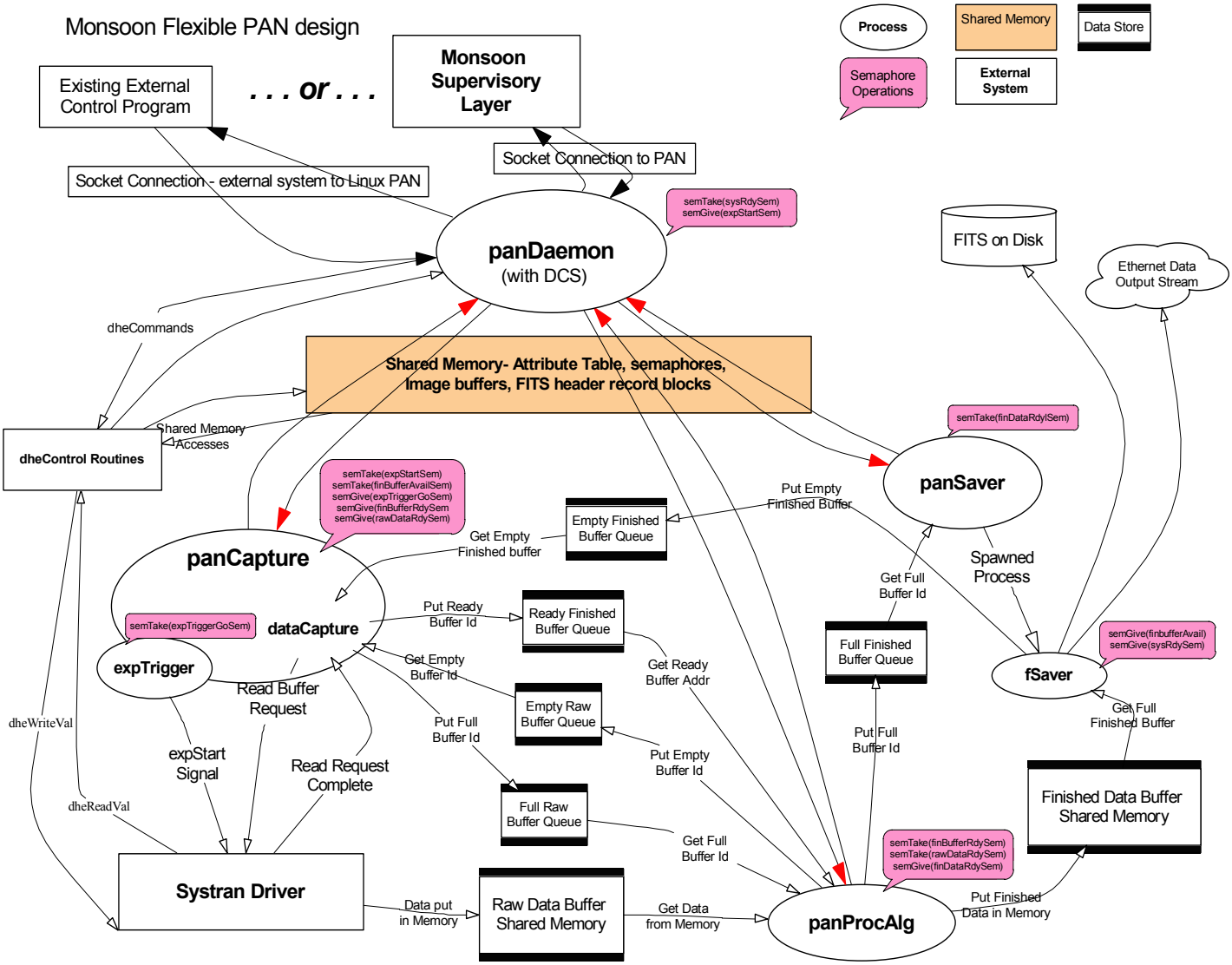
PAN Software Design

The diagram below is designed to explain the communications mechanisms used to coordinate the data processing tasks in the PAN software. A separate mechanism and diagram is used to send commands down to the DHE to configure it into an appropriate state. The command communication to the DHE occurs in the panDaemon command processor. Only one command expStart is sent from the expTrigger process and is handled differently from all others in the DHE. (See the detailed design document "MONSOON Image Acquisition System (Pixel Server) - Pixel Acquisition Node - Software Design Document, ####.9999.#a for a complete explaination of the process structure.)



PAN Process Design

This document outlines the design for the co-operating processes that make up the PAN software. There are four modules or primary processes in the system.

panDaemon - This module is a single process which initializes shared memory, semaphores and the systran board driver. It then spawns the other three primary processes and finally becomes a command processor that listens for input messages from the keyboard or a socket and executes the commands received. The primary communications method between this process and the others is the attribute Table in shared memory, the hiMem structure containing buffer pointers, queues & FITS header blocks and a number of semaphores which allow control to be passed between the processes. This process handles the expStart command that starts data flowing through the system. It must take the sysRdy semaphore before it can initiate an exposure by giving the expStrt semaphore.

panCapture - This is module consists of a pair of processes. The panDaemon spawns the initial process which initializes the pointers to shared memory spaces, acquires access to the required semaphores and does some initialization. It then spawns the other process and becomes the dataCapture process.

dataCapture - this process sits in a loop waiting to take the expStrt semaphore. This is given as a result of an expStart command line sent to the panDaemon command processing loop. It then from the emptyFnshdBuffer queue with its associated FITS header list and copies the FITS header list from the current list to the appropriate slot, puts the buffer index on the readyFnshdBuffer queue.

The process then loops taking a buffer from the emptyRawBuffer queue and when ready it gives the trigGo semaphore and waits for an appropriate amount of data to be sent by the DHE to the PAN. When the data has arrived the index of the raw data is put on the fullRawBuffer queue and the rDataReady semaphore is given. After which the process returns to the head of the loop

expTrigger - this process sits in a loop waiting to take the trigGo semaphore given by dataCapture. when it gets the semaphore it waits a short time (50 ms??) for the dataCapture process to finish setting up the data read and then sends the appropriate signal down to the DHE to begin the data generation. The captMode flag determines the appropriate signal and can contain three possible modes, NORMAL indicates a normal image transfer of a number of pixels equal to numRows*numCols. XMIT indicates a number of pixels generated by the MDB XMIT FPGA equal to tSize. ACQBD indicates a number of pixels generated by an Acquisition board FPGA equal to tSize. For an ACQBD image we always send 2048x2048 pixels.

panProcAlg - This is module consists of a single process which initializes the pointers to shared memory spaces, acquires access to the required semaphores and does some initialization. The process then sits in a loop taking a finished data buffer from the readyFnshdBuffer queue and then entering the processing loop.

In the processing loop the process waits to take the rDataReady semaphore. It then gets the next buffer from the fullRawBuffer queue and processes the buffer in accordance with the currently active processing algorithm. When the process if done with the raw buffer it is placed on the emptyRawBuffer queue for reuse.

When all processing is done for the current image, the processed data buffer is placed on the fullFnshdBuffer queue and the fDataReady semaphore is given.

panSaver - This is module consists of a single process which initializes the pointers to shared memory spaces, acquires access to the required semaphores and does some initialization.

The process then sits in a loop waiting to take the fDataReady semaphore. When the semaphore is taken the process removes a buffer from the fullFnshdBuffer queue and spawns an fSaver task. Once the task is spawned the process takes the bufferAvail semaphore before giving the sysRdy semaphore.

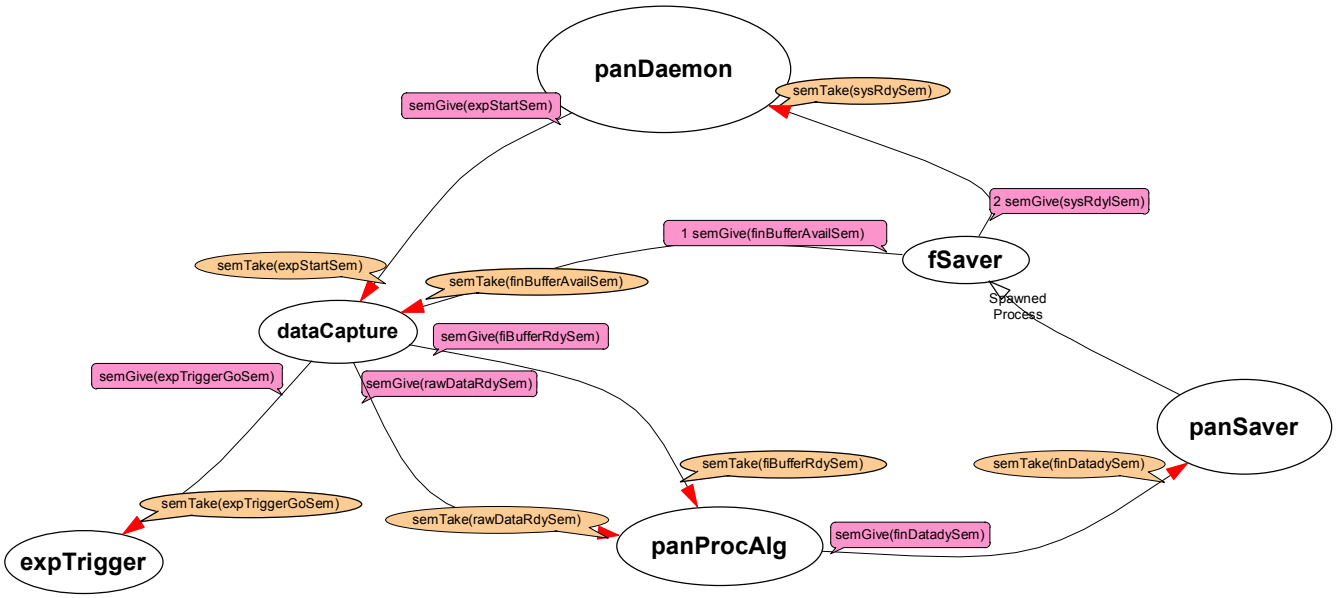
fSaver - these are transient tasks which copy data from the finished data buffers and the associated FITS header block lists to a fits file on disk or to the output data stream (100BaseT ethernet). When done these tasks empty the header lists, put the data and header buffers back on the emptyFnshdBuffer queue for reuse and then give the bufferAvail semaphore before deallocating themselves.

PAN Buffer Queues

emptyRawBuffer
emptyFnshdBuffer
readyFnshdBuffer
fullRawBuffer
fullFnshdBuffer

PAN Semaphores

Monsoon PAN Semaphores



sysRdy
expStrt
trigGo
rDataReady
fDataReady
bufferAvail