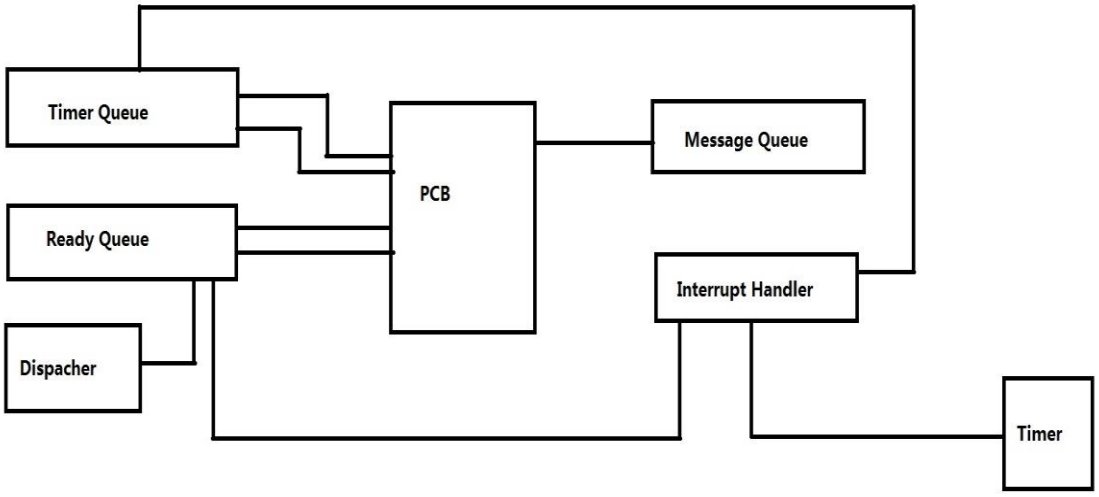
# Architectural Document

## What is included in my system

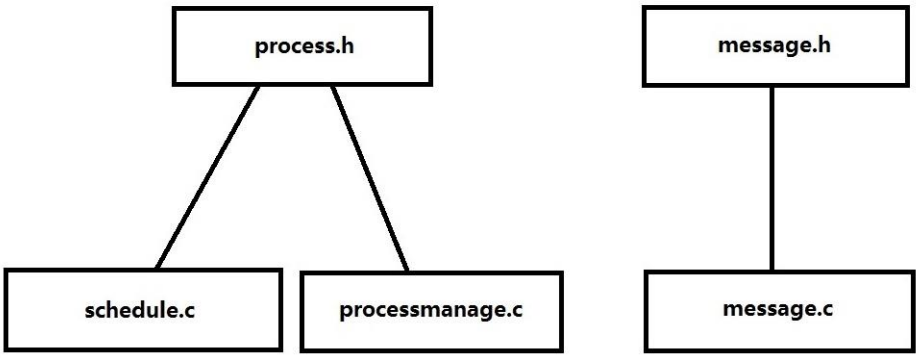
1. There are three main sections in my system:
   1.1 the process table
   1.2 the timer queue and ready queue.
   1.3 The message queue.
2. The features my system can do:
   2.1 create process
   2.2 call sleep to give up CPU actively.
   2.3 Terminate process
   2.4 A dispather in charge of finding the next ready process.
   2.5 Change process's priority.
   2.6 Suspend one process so it can not run until resume.
   2.7 Resume one process.
   2.8 Send one or all process a message.
   2.9 Receive from one or any process a message.

## High level design

1. Major module design



2. Major file hierarchy




3. Major routine flows

| **SYS CALLS** | SLEEP | CREATE PROCESS | TERMINATE PROCESS | SUSPEND | RESUME | CHANGE PRIORITY | SEND MESSAGE | RECEIVE MESSAGE |
|---|---|---|---|---|---|---|---|---|
| **Sub Routines** | Start_timer | AddToReady Queue | RemoveFrom ProcessTable | ClearReady Queue | AddTo ReadyQue ue | Order ReadyQu eue | Enqueue MessageQ ueue | Remove Message |
| | | | ClearTimer Queue | | | | | |
| | AddTo TimerQueu e | | ClearReady Queue | Dispatcher (if it terminate itself) | | | | |
| | Dispatcher | | Dispather (if it terminate itself) | | | | | |

# Justification of the High Level Design

1. PCB block design

   In implement of PCB block, I there are several domains that worth mention.

   a. The PCB block contains state, which indicates that the current state of the process. The available options are: SUSPEND, NONSUSPEND. When marked by NONSUSPEND, it means the current process is either in running state or in timer queue. (The ultimate design is to take off from the timer queue as soon as it has been suspended, but right now I just block it from entering ready queue.)

   b. PCB block contains pointers ptr_RNode and ptr_TNode, which record the address of corresponding node in either timer queue or ready queue. A process can't be in the timer queue and ready queue in the same time. By adding these two domains, whenever I want to operate particular action to the process, it can be performed in leaner time, rather than search a whole queue.

2. Timer Queue

   Timer queue is maintained in a waiting time increasing order all the time. Whenever Timer queue is modified, which means one of AddToTimerQueue, RemoveFromTimerQueue, and ClearTimerQueue is executed, a RefreshTimer function is called. In this way, we calculate the expect time that the first timer node in timer queue to be waken. And reset the system clock register with a new count down time units.

3. Ready Queue

   The ready queue is designed to be maintained in not only first come first serve order, but also make the process with higher priority first. Everytime we call dispatcher to get a latest ready process, it get the first one from the ready queue. Because the process with higher priority are always in the front, we schedule high priority process first, even if they came into ready queue later.

4. Lock

   We add lock on critical area like ready queue. Whenever ready queue is visited or modified, lock should be get first, otherwise blocking the process until it get the lock. The reason why lock is necessary is because when making change in, say ready queue, a interrupt is likely to occurs and move a process from timer queue to ready queue. In the z502 system, the interrupt handler is implemented in real thread, therefore there will be two executing flows going on in the same time, which may cause unexpected result.

5. Dispatcher

   Dispatcher is something in charge of making decision on which to run next. In some particular cases, like, both timer queue and ready queue are empty, as well as no suspended process, then call z502Halt, which shut down the system. In usual case, dispatcher get the first process in

ready queue and call switchcontext.

## Source code

Source code can be seen in file.

## Output

The output of project is attached on another file called Test Ouput. The output format is strictly on basis of giving format as flowing.

| Test name | Output Generated in test.c | SVC/Fault/ Interrupt | Scheduler Printer | Memory Printer |
|---|---|---|---|---|
| test0 | Full | Full | No | No |
| test1a | Full | Full | No | No |
| test1b | Full | Full | No | No |
| test1c | Full | Limited | Full | No |
| test1d | Full | Limited | Full | No |
| test1e | Full | Full | No | No |
| test1f | Full | Limited | Full | No |
| test1g | Full | Full | No | No |
| test1h | Full | Limited | Full | No |
| test1i | Full | Full | No | No |
| test1j | Full | Limited | Full | No |
| test1k | Full | Full | No | No |