DEC 4

This is a preliminary document describing the interrupt handling code at the command level in the new system.

Objective

This is designed to handle 2 kinds of interrupt.
major panic – break button,
minor panic – currently control shift p.

~~In a process~~

The action created by these panics is to appear as follows:

If they occur while user code is running

a) minor panic,

The user subprocess at top of the stack will receive an interrupt with status "internal minor panic". If no user ~~subprocess currently running~~ which is a proper or improper ancestor of the current top of stack user subprocess has interrupts enabled, the interrupt will be accepted by the bead ghost, which will call a debugger.

b) major panic

The bad ghost will be called, and will in turn call a debugger.

If they occur while system code is running, either called explicitly or implicitly (error and panic) by user code then:

each system routine in turn on the stack cleans up. If that system routine considers itself to be a command type routine, it informs the user try of the interrupt and requests a new command. (Thus stopping the clean up procedure.) (The debugger is such a subprocess.) If that system routine does not consider itself to be a command type routine, it removes itself from ~~the stack and dies~~ the stack.

(one way to cleanup is to do nothing, e.g. the low level disk system.)

if user code can call a command processor, the user should be able to instruct said code to return with (major minor) interrupt;

can user code explicitly invoke a CP? (it can implicitly, with an error.)

Fact*:

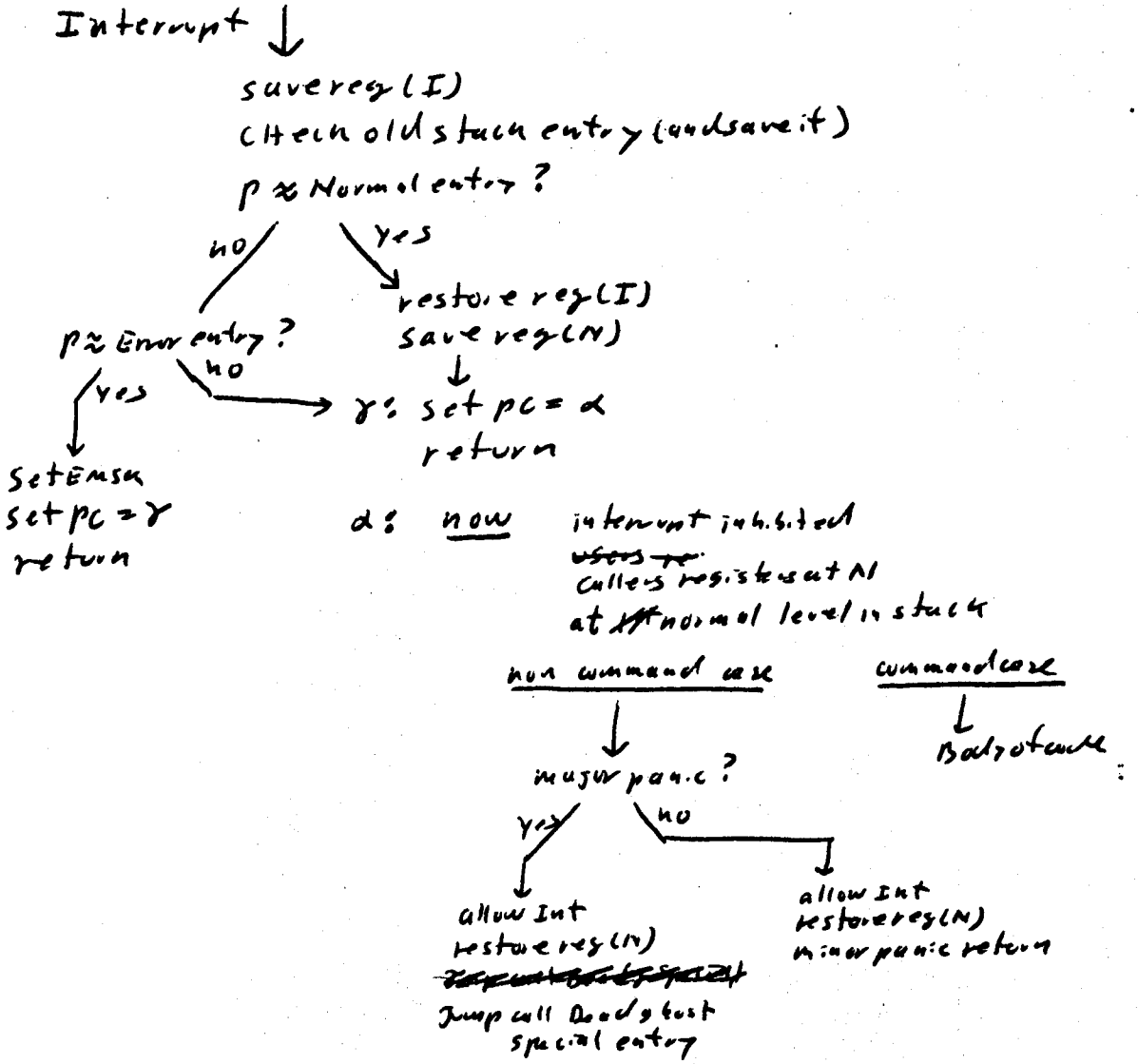
If a leaf system subprocess gets an interrupt or an error, it was on the stack immediately below.

Note:

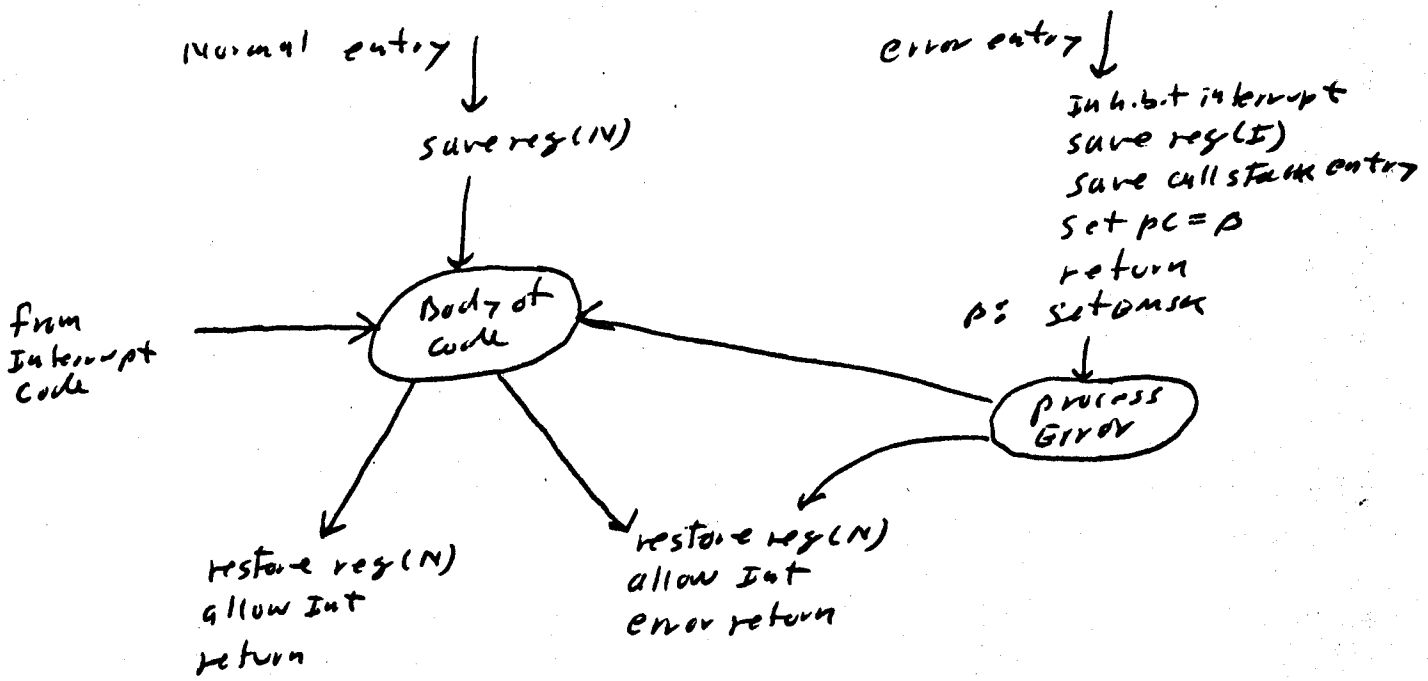
These algorithms use various portions of my latest proposal on the subprocess call, return and interrupt mechanism. In particular, at least inhibit interrupt in the subprocess descriptor, and the return with interrupt specification.

* unfortunately, this fact is [redacted] sometimes irrelevant.

System Leaf


Interrupt
saveregs(I)
Check old stack entry (and save it)
p ≈ Normal entry ?
no
yes
restoreregs(I)
saveregs(N)
p ≈ Error entry ?
no
yes
γ: set pc = α
return
SetEMSA
set pc = γ
return
α: now interrupts inhibited
Callers registers at N1
at normal level in stack
non command case
command case
Bad stack
major panic ?
yes
no
allow Int
restoreregs(N)
Jump call Dead g host special entry
allow Int
restoreregs(N)
minor panic return

System leaf continued



System leaf continued

How to send and get events and resume

send

inhibit interrupt

sendev (do not have a p-counter offset error)

set flag for sent

allow interrupts

get

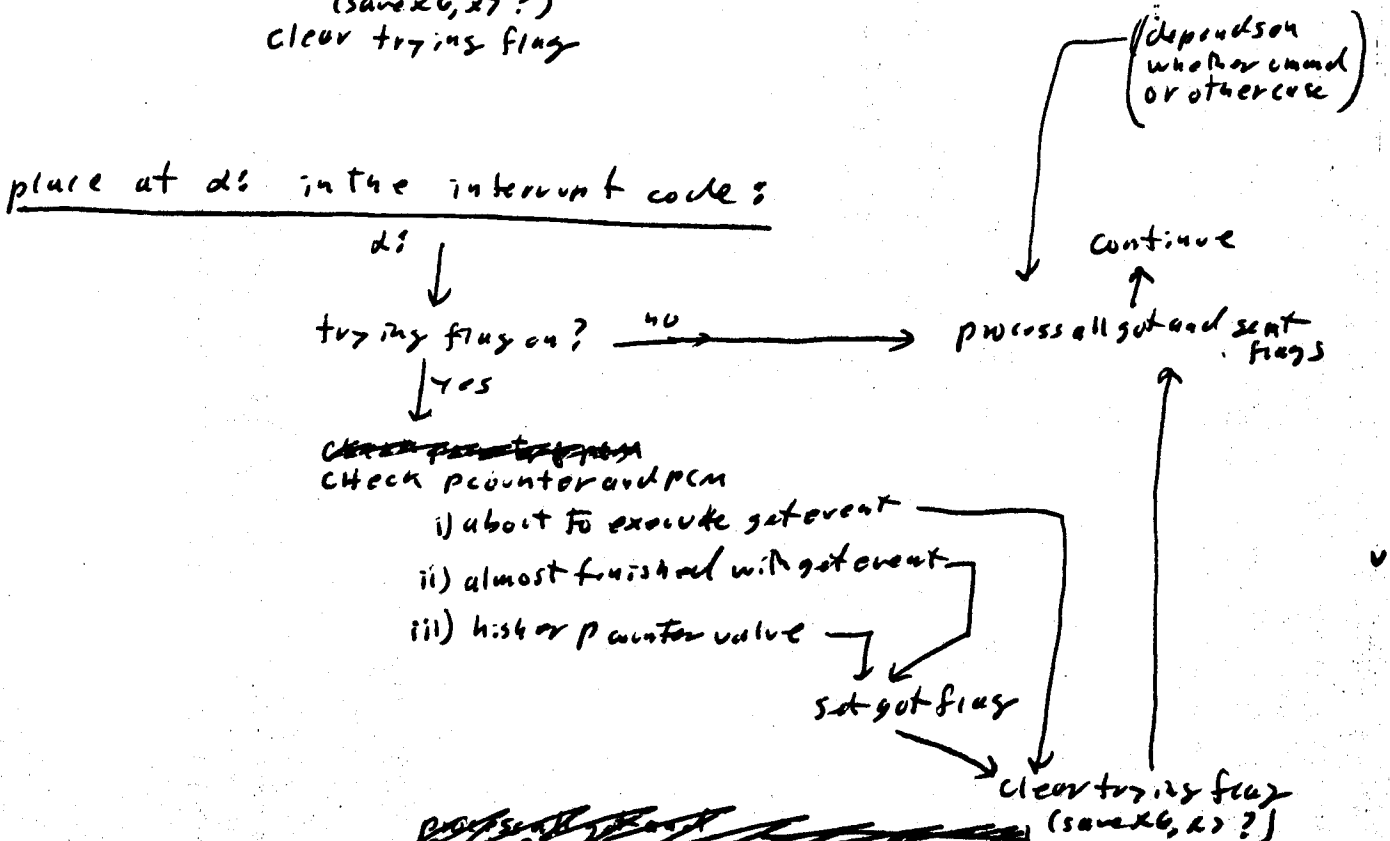
clear got flag? ← set trying flag

getevent or hang

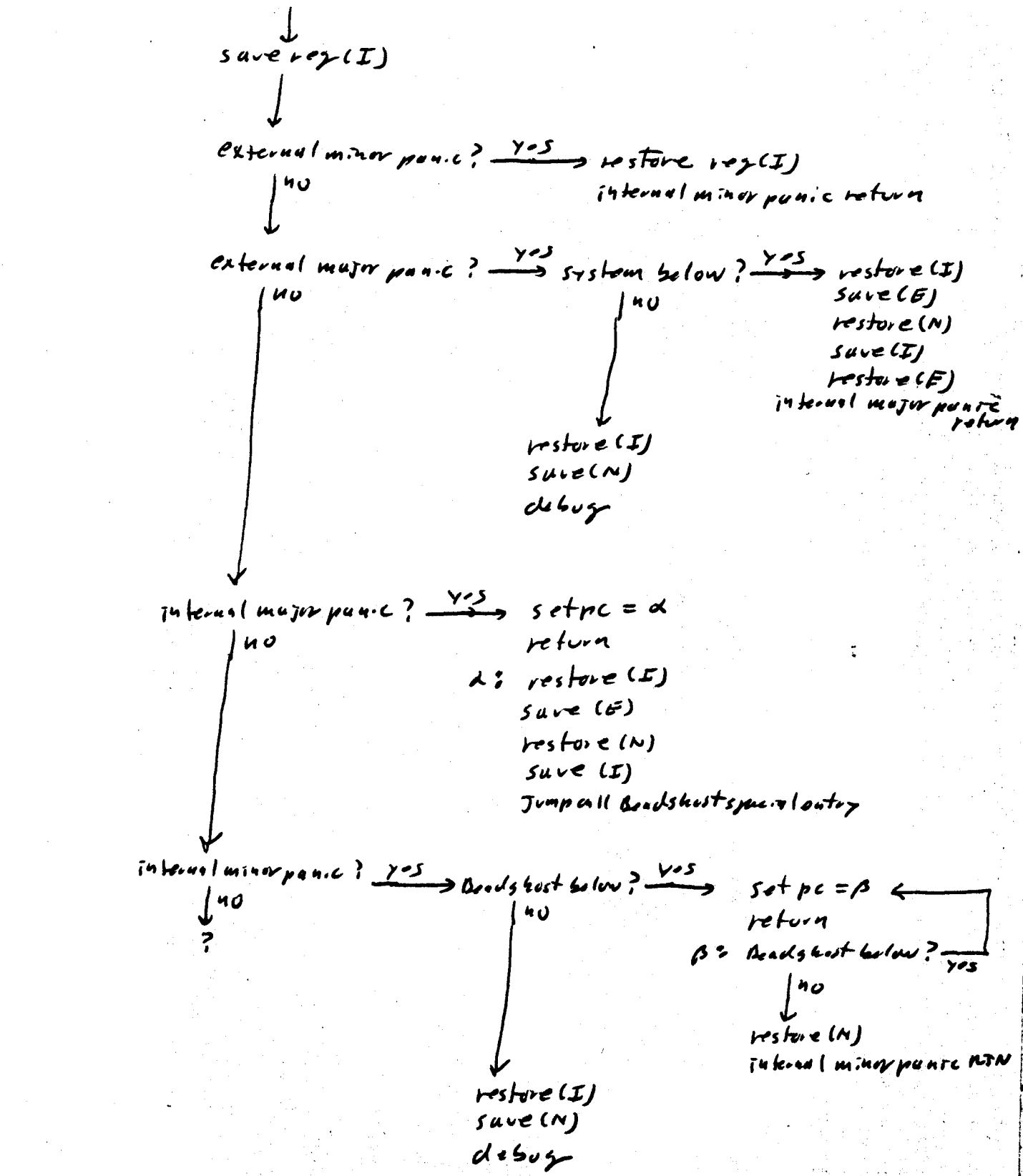
set got flag (save x6, x7 ?)

clear trying flag

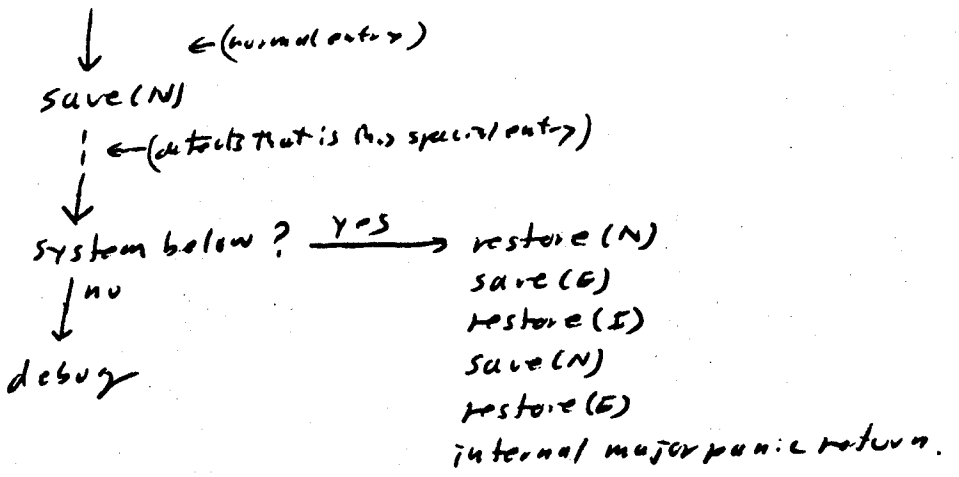
place at d: in the interrupt code:



Beadghost interrupt entry



Bread ghost special entry



note! The bread ghost has interrupt inhibit on at all times