INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Regulation of Life History Strategies Within Individuals in Predictable and Unpredictable Environments

by

Jerry Dale Jacobs

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

1996

Approved by John C. Wingfield
(Chairperson Of Supervisory Committee)

Program Authorized
to Offer Degree Zoology

Date 8/14/96

In presenting this dissertation in partial fulfillment of the requirements for the Doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to University Microfilms, 1490 Eisenhower Place, P.O. Box 975, Ann Arbor, MI 48105,to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature Jerry Jacobs

Date 10/14/96

University of Washington

Abstract

Regulation of Life History Strategies Within Individuals in Predictable and Unpredictable Environments

by Jerry D. Jacobs

Chairperson of the Supervisory Committee: Professor John C. Wingfield
Department of Zoology

Animals must adjust their physiology and life histories to conform with changes in the environment. These changes can be fairly predictable, such as the changes in the season, or they can be unpredictable, such as an attack by a predator or an unusual storm. A finite state machine model was developed to understand individual responses to changing environments. The model was used to investigate life history diversities, and changes in animals associated with unpredictable environmental events. Several conclusions were drawn from the model with regards to unpredictable events: Animals in very predictable environments, very unpredictable environments, and animals in constrained environments would not show the normal adrenocortical responses to unpredictable events. The predictions were tested with experiments, and follow the general trend of the data present.

Table of Contents

List of Figures

List of Tables

Acknowlegements

I would like to express my thanks to all those who made this dissertation possible. My committee (Aimée Bakken, Walt Dickhoff, Alan Kohn, Ray Huey, and John Wingfield) provided valuable insights and suggestions for the work. I am especially grateful to my advisor, John Wingfield, for his unflagging enthusiasm, generous support, and persistent encouragement. My special thanks goes to those people who were hardy enough to brave the wind, rain, and cold nights or immersion in cold water involved in collecting blood samples the fishes: Laura Hurley, Stuart Cole, Lee Astheimer, Sara Wooley, Tom Hahn, Wai Pang Chan, and my wife Stephanie Pino-Heiss. Lynn Erckmann provided good-humored help with the steroid assays, as well as making the lab an enjoyable place to work. I wish to thank a number of people for their willingness to share ideas, and helping to make graduate school more pleasant: Marilyn Ramenofsky, Ken Morgan, Steve Schoech, Gene Fowler, Rachael Levin, Michael Romero, and Cathy Pfister. My fondest thanks go t my wife Stephanie Pino-Heiss for putting up with my ill-humor while preparing this dissertation, as well as helping in several of the lab and field experiments.

Chapter I:

Using a Finite State Machine Model to Understand Organismal Responses to Changes in the Environment.

Introduction

No free-living organism lives in a truly constant environment. All species are exposed to fluctuations in environmental conditions, some of which may be extreme. For example, in arctic environments, temperature, food availability, and the background color of the environment vary radically in a yearly seasonal cycle. In some tropical regions, the dry and rainy periods occur at specific times of year. Even in places that at first seem exceedingly stable, the environment fluctuates. In caves, where the yearly temperature may only fluctuate on the order of 2 C°, seasonal flooding may occurs, providing changes in the nutrient input from non-cave sources (Poulson, 1964). Some cave animals get their food from nematodes within bat feces, or on the fungus which on the guano. Thus there may are seasonal variations in food availability within the cave due to seasonal changes in insect populations (food for the bats) outside, and changes in the activity level of the bats (Chapman, 1993).

Unless the environmental fluctuations are very small, an organism will have to modify its physiology to survive as conditions change (*e.g.* Prosser, 1986). In this dissertation, the term physiology is used in its broadest sense to include all the

biological properties of the organism, including morphology and behavior (Eckert et al., 1988). Physiology can be adjusted several ways to maximize survival in a fluctuating environment. Some invertebrates use a developmental strategy to adapt to long range, predictable changes. The phenotype of the adult is determined by changes in the development pathway, leading to adults with appropriate physiology (*e.g.*, Tauber *et al.*, 1985 and Mustafa and Hodgson, 1984), or individuals that will enter diapause in order to overwinter (Hairston, and Olds, 1986). In others, the presence of predators triggers developmental switches inducing defensive phenotypes. Developmental changes are not reversible within an individual, and are often determined before fertilization. Developmental switching of phenotypes is very successful for some animals such as rotifers (Gilbert, 1980), *Daphnia* (*e.g.* Sibor, 1992), and bryozoans (Harvell, 1992). These animals have life cycles much less than a year, and can respond to changes in the environment by adjusting the physiology of its progeny, or permanently changing their own morphology. In these short lived organisms, the temporal scale of environmental changes is *greater* than the generation time of the organism. Other organisms, particularly long lived ones, have generation times and life spans that may be many times greater than the temporal scale of environmental change. They show dramatic changes in physiology within the individual, and these must be reversible, or cyclic so as to acclimate to repeated environmental fluctuations (Prosser, 1986). These latter responses will be the focus here.

Phenotypic Expression and Life History Stages

Richard Levins (1968) described several strategies of adaptation in a changing environment. If a fitness curve for a phenotype is plotted against a gradient of some limiting environmental variable, a more or less bell shaped curve results (Figure 1.1a). This curve need not be symmetrical (and more often than not will not be), but for simplicity we will use symmetrical curves in this discussion. If an organism were not limited in any way, an infinitely tall and infinitely wide curve would result so as to maximize fitness under all situations (unconstrained phenotype, Figure 1.1b). Clearly, this is not possible, as the physical and biological composition of an organism limits it, so that an animal (or plant) can not be perfectly adapted to all niches. An ant, for example, no matter how well adapted to living in an *Acacia* bush, would probably not do well in a swamp, and certainly could not penetrate the temperatures, pressures, and aquatic nature of the habitat at deep sea hydrothermal vents. Thus organisms have constrained phenotypes. The height and width of the fitness curve is limited by the biological and physical structures of the animal's genotype. For any given phenotype, the fitness curve will have an optimum for that environmental variable, and regions where the fitness is near or equal to zero. (Figure 1.1a)

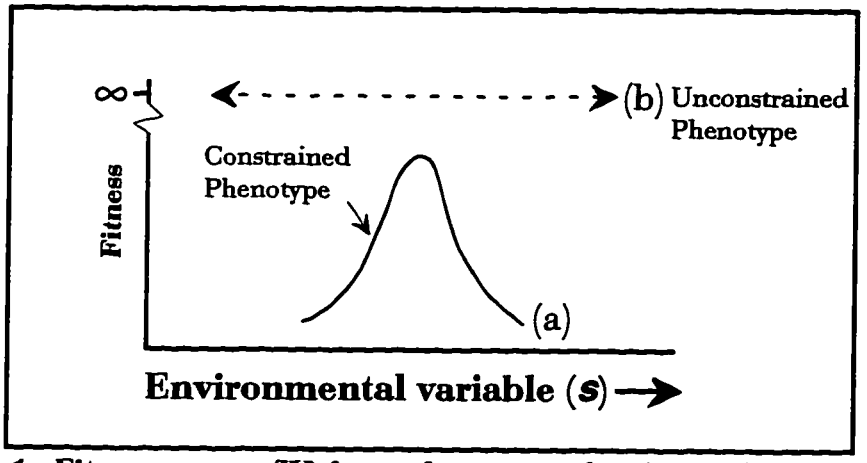


Figure 1.1 - Fitness curves (W) for a phenotype, showing a phenotype which is constrained by its physiology/environment (solid line) and one which is neither contrained by its physiology nor by the environment (dotted line).

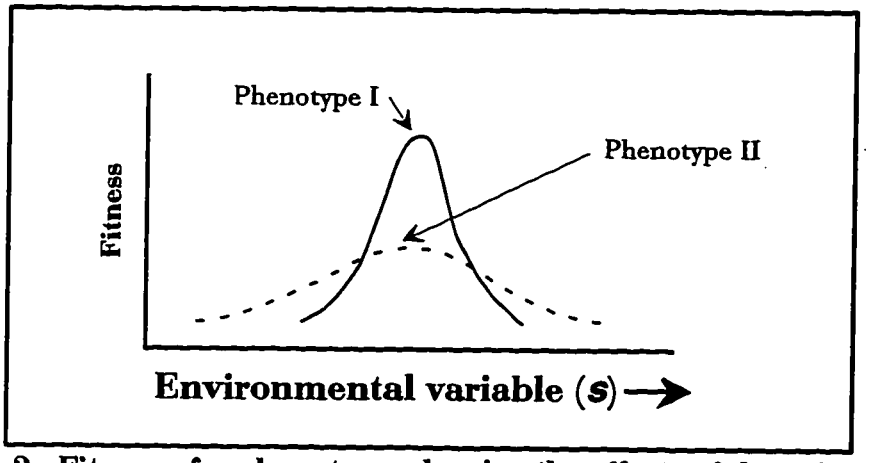


Figure 1.2 - Fitness of a phenotype, showing the effects of the principle of allocation. Since the area under the curve is limited by the genotype, if the curve is broadened (adaptable to a broader range of the environmental variable) then the maximal fitness is then reduced.

Levins (1968) asserted the principle of allocation, which basically states that an organism has limited resources, and that these resources must be allocated between different demands on the organism such that:

$$\int W(s)ds \approx C$$

where C is a constant, s is some environmental variable (such as temperature, salinity, hydrostatic pressure....), and W is the fitness curve for the phenotype. Because the shape of the curve is determined by the intrinsic limitations of the species, the dotted curve which is broader (phenotype II, Figure 1.2), must have a lower peak than the narrower curve (phenotype I, Figure 1.2) and thus reduced maximum fitness. The out come of this principle is that if a phenotype is adapted to a broader range of the environmental variable s, the maximal fitness for the phenotype will be less, because the area under the fitness curve is assumed constant (See figure 1.2). Both the dotted and solid curves have the same area under them. In biological terms, we interpret this as follows: a phenotype adapted to a broader range of s, will either require a larger number of enzymes and receptors, etc. that must be expressed simultaneously even though they may only be used intermittently, or enzymes which are functional over a broader range of s, but have lesser maximal rates (Somero, 1978 and 1995). These clearly would incur greater energetic costs, or reduced maximal efficiency compared to a phenotype adapted to a narrower range of s and thus only expresses enzymes and receptors specific for that narrow range. Enzymes specialized for a narrow range

of an environmental variable usually have higher maximal rates than those adapted for broader rates, though this is not always the case (Hochachka and Somero, 1973).

Organisms can "cheat" the allocation principle by being able to express multiple phenotypes within the same genotype during successive generations (see Stearns, 1989). These phenotypes are optimized to particular niches within the environment that the genotype experiences over time and space (*e.g.* Gilbert, 1980). Such phenotypic responses can be a limited number of polyphenisms which are designed to cope with specific problems in the environment such as predators and overwintering, or they may be a continuum of phenotypes which develop in direct response to the environment (norms of reaction - Schmalhausen, 1949; Stearns and Koella, 1986). Let's examine a simplified environment which varies along a single environmental variable s, which could be temperature, salinity, etc. Assuming that the organisms have some means of anticipating changes in the environmental variable, switching between the 2 different phenotypes from one generation to the next will provided a higher fitness for the genotype. This fitness gain hinges on a number of factors. The generation time for the organism must be less than the time between the extremes, as if the generation time is longer than the period of the environmental variation, the organism can not produce progeny which match the environment, as they reproduce too slowly. The cost of switching phenotypes combined with the cost of maintaining the

genetic components necessary for the switching must be less than the gain in fitness from the switching. If the fitness cost of the phenotypic switch is too high, it will exceed the fitness gained by the switch. The changes in the environment must have a predictable component, because if the environment were totally chaotic, it would be impossible to time the correct progeny with the proper environment, and thus a generalist (mixed) phenotype would be superior. The genotype must be able to survive the periods where the fitness is very low. If a genotype has two phenotypes optimized for the extremes of its environment, and environmental variation is large, there would be a region of the environmental gradient where the fitness is very low (Figure 1.3c). If this region of reduced fitness represents a period where conditions are unfavorable for reproduction, and is of short duration, then this low fitness region poses no problem. If the reduced fitness is due to impaired survivorship, then the genotype will die off during the "zero-fitness" period as the environment switches between the extremes.

For an environment with only a very small variation in s (Figure 1.3a), two alternate phenotypes would be very similar, and switching between them would probably not provide any advantage over expressing a generalist phenotype with only slightly reduced maximal fitness (dotted line, Figure 1.3a) due to costs inherent in switching phenotypes. As the variation in s becomes larger, the difference between the maximal fitness of the generalist phenotype (dotted line, Figure 1.3b) and phenotypes I and II (solid lines, Figure 1.3b) becomes larger and begins to offset the switching cost. If the variation in s is moderate to large

(Figures 1.3 c,d), and switches predictably between the extremes in s, the genotype may not be able to persist when the environment is in the middle range for the variable (i.e. the region of the variable where if the organism is forced to complete a generation, the fitness is very low or zero [see Figure 1.3d]). Generalist phenotypes now have greatly reduced maximal fitness. In such a situation, a third phenotype (or even more) would be required to fill the gap (Figure 1.4a). If the cost of switching phenotypes is small and the generation time is small, the organism can smooth out the overall fitness curve for the genotype, by having a larger number of similar phenotypes (Figure 1.4c). Note again that the generalist phenotype would have greatly reduced maximal fitness (dotted line, Figure 1.4a,b,c). Expression of many different phenotypes from a single genotype (as in Figure 1.4c) would probably result in higher costs than advantages gained in fitness. This example would be advantageous only if the organism lived in both an extremely predictable environment, so that the timing of the expression of appropriate phenotypes could be precise, and that the organism had a very short life cycle in order to map many phenotypes with corresponding changes in the environment. In an unpredictable environment, an organism can take the "shot-gun" approach to producing young adapted to the environment, described by Kaplan and Cooper (1984). In this model, a variety of different phenotypes are

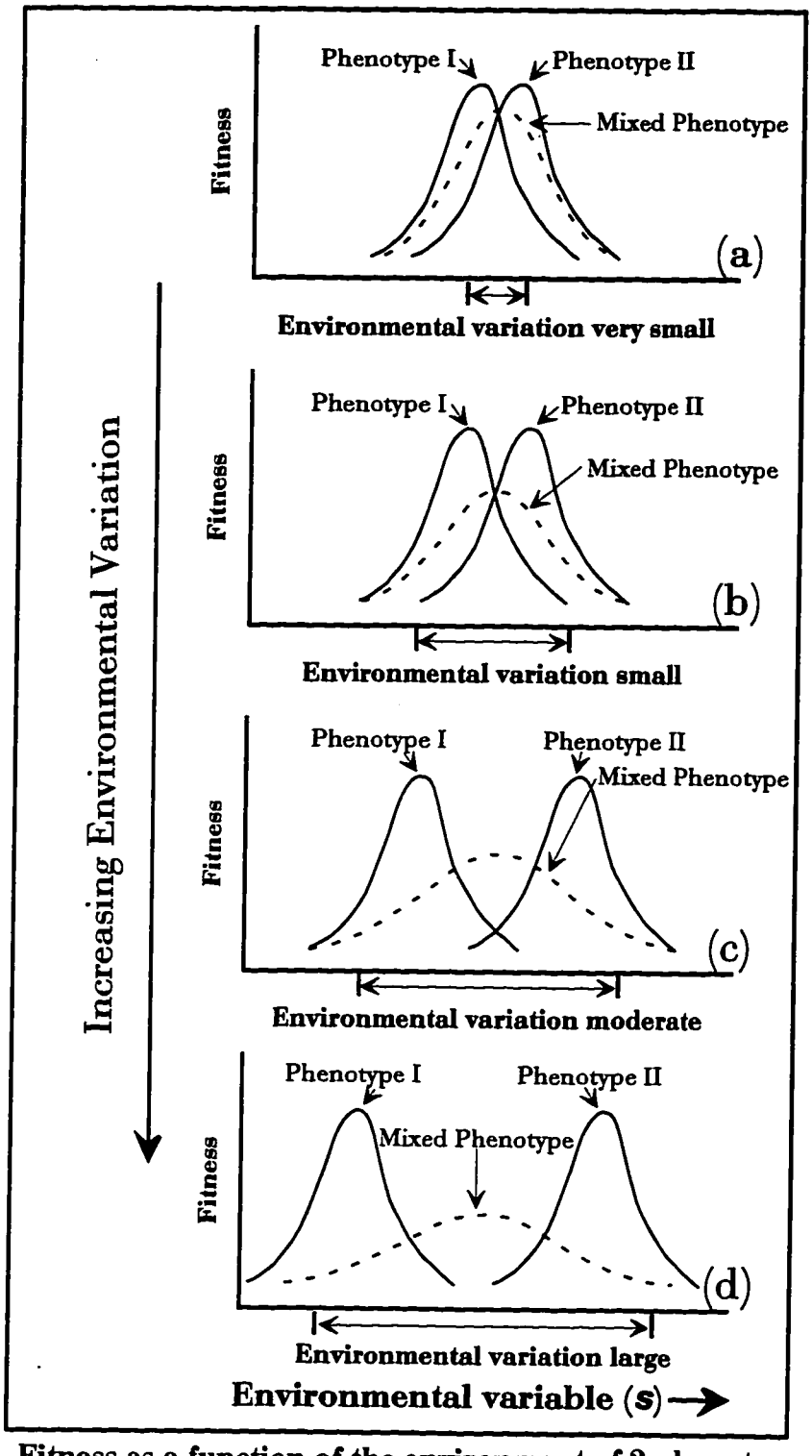


Figure 1.3 - Fitness as a function of the environment of 2 phenotypes from the same genotype. The dotted curve represents the fitness of a mixture of the two phenotypes (a generalist phenotype). If the environmental variations is small, the mixed phenotype will have a higher fitness over all fluctuations in the environment.

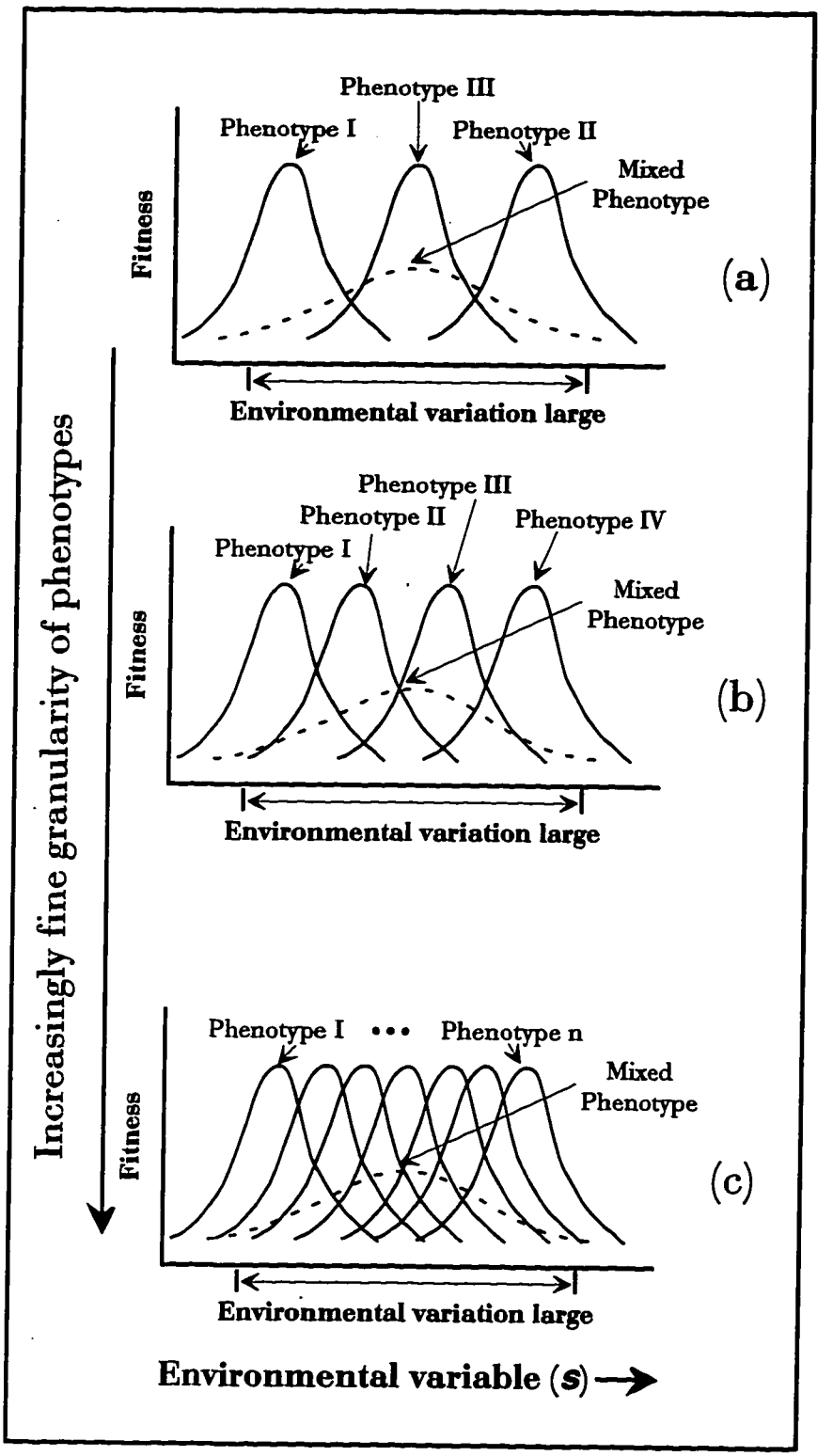


Figure 1.4 - Fitness as a function granularity of phenotypes. The dotted curve represents the fitness of a mixed phenotype (a generalist phenotype). As the granularity of the phenotypes becomes finer, the overall fitness for an animal.

produced such that the probability of some offspring surviving in the changing environment is increased.

Life History Stages Within Individuals

Most vertebrates live at least one year. Within the period of a single life cycle, environmental conditions will probably fluctuate (cycle) many times. This makes the strategy of "hard-wired" phenotype switching between generations untenable and the polyphenism type of phenotypic plasticity common in invertebrates is not found in vertebrates. This chapter is not intended as a review of phenotypic plasticity within populations (well developed by Stearns, 1989, and West-Eberhard, 1989), but rather to develop a model for physiological changes *within* an individual during its life cycle.

An individual must be able to survive a wide variety of environmental fluctuations. It does this by regulating gene expression to adjust morphology, physiology, and behavior to track or anticipate environmental changes (Prosser, 1986). The organism will often (particularly in temperate or arctic species) pass through a series of *phenotypic stages*, each appropriate for a part of the year, and the functions required at that time of year. *Phenotypic stages* are similar to the phenotypic changes described above, but they occur ***within*** an individual. For example, many animals undergo changes in coloration to adapt to seasonal changes in the color of the habitat or for thermoregulation. In the snowshoe hare, *Lepus americanus*, in the winter, the fur is white to match the snow cover and is

brown in summer to match the exposed forest floor. In spring and fall, as the hares molt between winter and summer colorations, a mottled brown and white pelage results which matches the scattered snow cover during these seasons (Ingles, 1947). Another striking example is the willow ptarmigan, *Lagopus lagopus*, which undergoes similar changes in plumage coloration on the arctic tundra (Pielou, 1994) and these will be addressed later (See Figure 1.6). Other examples of phenotypic stages include adaptations to different temperature ranges as in the goldfish (*Carassius auratus*; Fry *et al.*, 1942; Guderley, 1990) and the goldfinch (*Carduelis tristis;* Buttemer, 1985), or physiological changes associated with life history processes (*i.e.*, reproduction and molting). The temporal progression of phenotypic stages is driven by time and often in relation to changing seasons. During a complete life cycle, an animal will pass though a number of discrete stages which are species specific. These stages represent major events in the life cycle, such as development, breeding, migration, and molting. They can overlap, but usually are temporally separated and sometimes are mutually exclusive.

For further discussions, a few terms need to be defined. *Phenotype* (as used in this discussion), when used to refer to vertebrates, means the sum expression of all genes and stages within a given individual. A number of *phenotypic stages* corresponding to the different suites of extended phenotypes in the various life history stages exist within each *phenotype*. These *phenotypic stages* can represent transient conditions such as breeding and migration in birds, or they may be more

permanent changes, such as sex reversal in sequentially hermaphroditic fishes. Again, the progression of these *phenotypic stages* is analogous to the switching of phenotypes described above, only the "switch" occurs within an individual.

This leads to the logical extension of Levins' (1968) phenotypic models to *phenotypic stages* (Figure 1.5). In this extension, the *phenotypic stages* take the place of the phenotypes, and because we are now referring to individuals, rather than a population, we will be plotting a *fitness probability*, rather than fitness (since an individual can only have one fitness, at least as represented by progeny in the next generation). The *fitness probability* reflects both the animal's potential for producing offspring, and its potential survivorship. If the cost of transition between *phenotypic stages* is relatively small, the overall fitness probability for the individual can be increased by expressing a greater number of phenotypic stages. Again, as in the polyphenism, the transition of *phenotypic stages* requires a predictable component in the environment so that individuals can anticipate change and activate the appropriate *phenotypic stage*. It would be impossible match a *phenotypic stage* to the environment, if the environment were changing in a completely random manner, and any change in the *phenotypic stage* would more than likely be inappropriate. For a population, the "shot-gun" approach to producing offspring to match a changing environment (Kaplan and Cooper, 1984) can be used, but this is not an option for an individual.

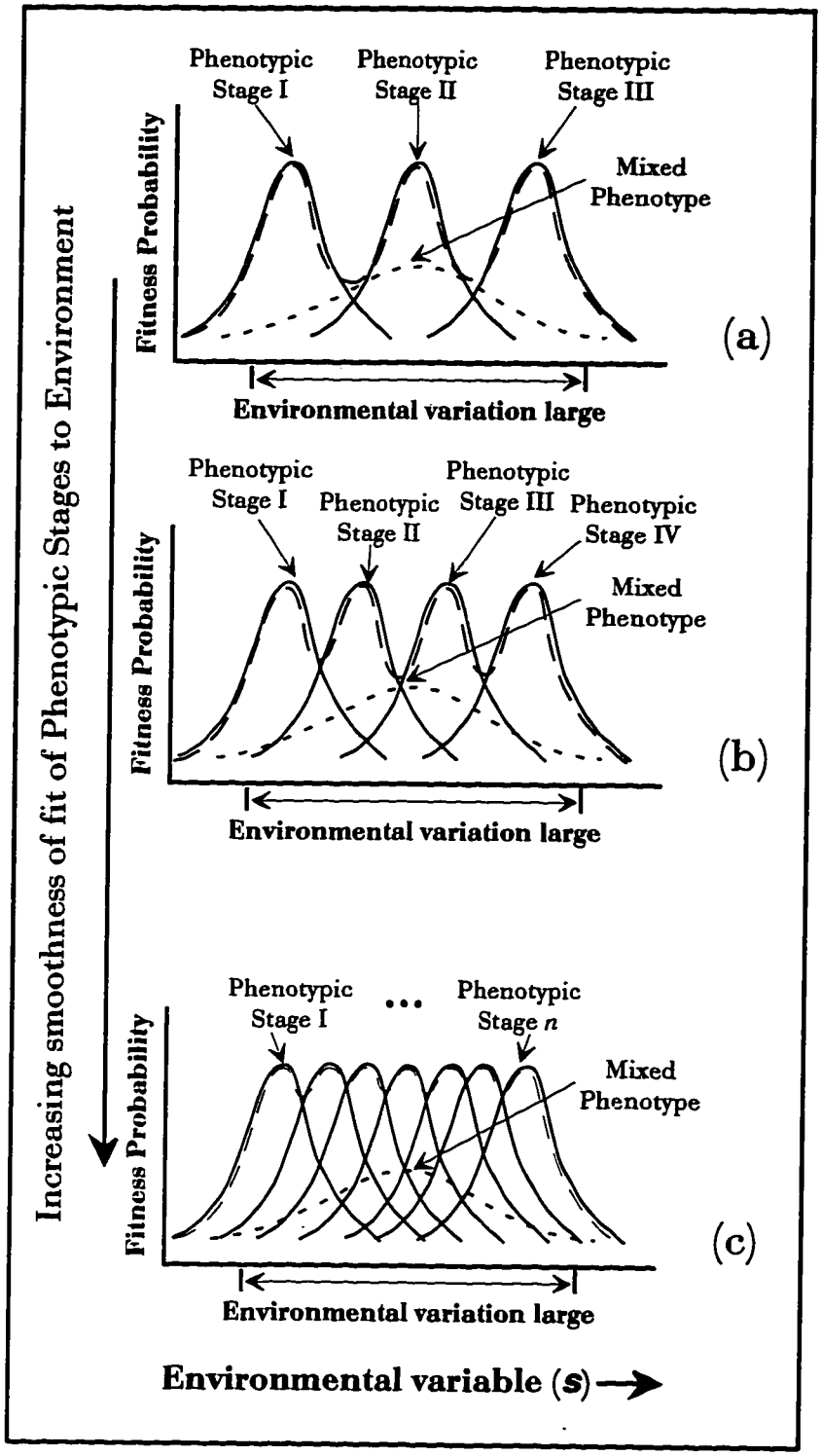


Figure 1.5 - Fitness probability as a function of granularity of phenotypic stages. The dotted curve represents the fitness probability of a mixed phenotype (a generalist phenotype) which is attempting to cover the entire environmental range. As the granularity of the phenotypic stages becomes finer, the overall fitness for an organism increases dramatically.

The willow ptarmigan (*Lagopus lagopus*) is a good example of the relation between progression of discrete phenotypic stages, and a generalist phenotypic stage that tries to adapt to the entire suite of environments simultaneously. Normally, the willow ptarmigan goes though a number of phenotypic stages including discrete molt patterns (Pielou, 1994; Figure 1.6a). During the winter, the bird is entirely white, to match the complete snow cover at that time. In spring, the head region is molted to match the patchy dark areas where the snow is melting through to the ground below. In summer, the ptarmigan completes the molt, and with the wings tucked in, the mottled pattern blends in well with its background. In fall, the molting from the mottled summer phase, into the white winter phase, yields a 'peppered" phase which blends well with the streaky pattern of the first snows in the fall. If the ptarmigan were to adopt a generalist phenotypic stage intermediate to the four normal phenotypic color stages (trying to cover the entire range of ground colors/patterns, Figure 1.6b), the resulting generalist molt would be inappropriate for almost all conditions the animal was exposed to. A ptarmigan expressing this generalist phenotypic stage would probably be quickly removed from the gene pool by predators. The ptarmigan generation time is too long to produce new individuals for each environmental phase. Furthermore, breeding is only possible during one part of the year, further limiting the possibility of generational responses to seasonal fluctuations in the environment.

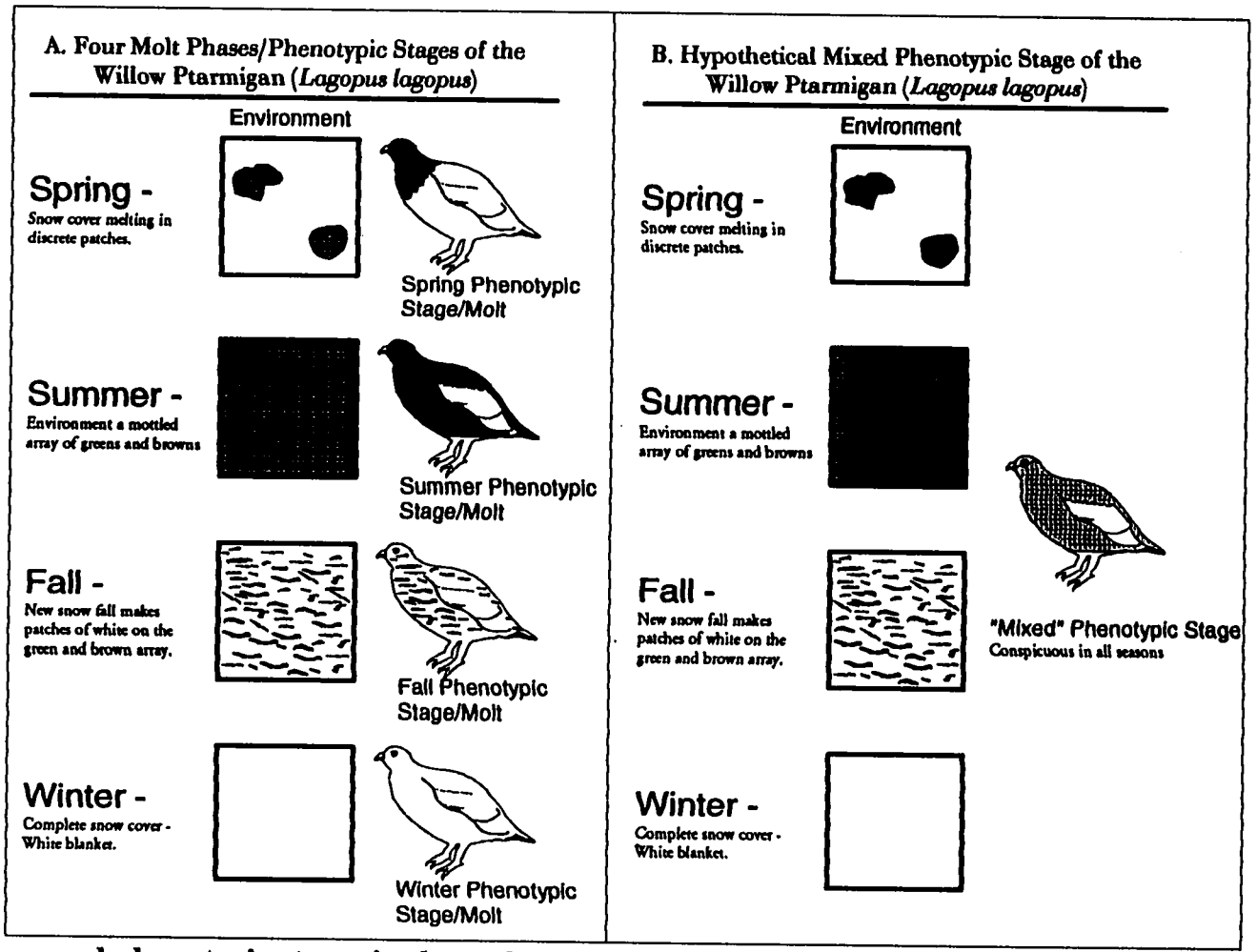


Figure 1.6 - Seasonal phenotypic stages in the molt cycle of the willow ptarmigan,Lagopus lagopus, and the ground cover for the corresponding season. (A) Phenotypic stages found in wild willow ptarmigans. (B) Hypothetical 'mixed' phenotypic stage which attempts to take a 'middle-of-the-road' approach to adapting to seasonal changes in the ground cover.

The discussion thus far has assumed that the changes in environmental conditions are more or less predictable. Transition either between phenotypes in a population, or among *phenotypic stages* within individuals must be timed so as to coincide with appropriate environmental changes. In either case, information from the environment (environmental cues; Levins, 1968) permits organisms to anticipate the change. The importance of these cues is the topic of the next section.

Environmental Cues

In an environment with predictable components, each *phenotypic stage* shows a progression of events, driven by cues from the environment. These cues comes in five basic classes: developmental, initial predictive, supplementary, synchronizing and integrating, and modifying factors (see also Wingfield and Kenagy, 1991).

1) Developmental cues come in several different types: growth factors/morphogens, tissue interactions, intrinsic factors, and external factors. They most often come from the internal environment of the embryo, but some cues come directly from the external environment (i.e., extrinsic factors). Growth factors/morphogens (such as fibroblast growth factor [FGF] and Retinoic acid) trigger/regulate the differentiation of tissues and structures (limb development, differentiation of muscle cells...). Both the presence and absence of growth factors can act to stimulate differentiation. When FGF is removed

from cultures of dividing myoblasts, they differentiate and begin to fuse into myotubes (Linkhart, Clegg, and Hauschka, 1980). During vertebrate limb development, a gradient of retinoic acid is set up which regulates differentiation and pattern formation of the limb (Eichele, 1989). Tissue interactions can act as signals for differentiation. Development of the optic vesicle stimulates the overlying ectoderm to differentiate into a lens (Jacobsen and Sater, 1988). Intrinsic factors, such as the nuclear/cytoplasmic ratio can signal differentiation events. In developing *Drosophila melanogaster* embryos, the formation of the cellular blastoderm is stimulated by a critical nuclear to cytoplasmic ratio (Edgar, Kiehle, and Schubiger, 1986), and in embryos of the African clawed frog, *Xenopus laevis*, the mid-blastula stage transition (where nuclear genes first become activated) is also regulated by a critical nuclear-cytoplasmic ratio (Newport and Kirshner, 1982a, b). External factors, such as temperature in reptiles (Janzen and Paukstis, 1991) and fish (Conover and Fleisher, 1987) and pH in fish (Schaefer and Rubin, 1985) can shunt a developing animal into a different path of development. In some freshwater teleosts, the sex of the fry can be all male if the eggs are developed in low pH water, and all female if developed in high pH water (Schaefer and Rubin, 1985). In some reptiles, such as alligators (Ferguson and Joanen (1983) and some turtles (Lockwood *et al.*, 1991), the sex of the embryo is determined by the temperature at critical periods during development (*e.g.* Crews, *et al.*, 1994).

2) Initial predictive cues are environmental signals that allow the organism to make long term predictions about the environment. Photoperiod is a good example of a cue which provides information on future events in diverse environments. Increasing day length is an excellent predictor of the approach of spring and summer, and triggers gonadal development in anticipation of the breeding season in many species (Follett, 1984; Gwinner, 1986).

3) Supplementary cues (such as food availability) provide fine tuning for responses to initial predictive cues, and also provide short term predictive information about the environment (Marshall, 1959; Wingfield and Kenagy, 1991). As the initial predictive cues only provide a general idea of what the environment might be, supplementary cues are important to ensure the organism makes the correct responses at the correct time. In other words, they can accelerate or inhibit the responses to initial predictive cues.

4) Synchronizing and integrating cues are social stimuli provided by other members of the population (Hinde, 1965; Lehrman, 1965). They ensure that the animal's responses to the external environment are appropriate to the condition of the other conspecifics. For example, the sea star *Acanthaster planci* releases its eggs and sperm into the ocean where they are fertilized. If males and females spawn at the same time, over 70% of the eggs are fertilized up to 8 meters downstream of a single spawning male and greater than 5% even 100 meters downstream (Babcock et al., 1994). Fertilization would be

exceedingly unlikely if spawning was not synchronized as the sperm and eggs would become diluted in the vast volume of sea water, and the shed eggs and sperm would thus be wasted.

5) The final class of cues/factors are the modifying factors or transient perturbation factors (TPF or "stressors"). Unpredictable events in the environment (*e.g.* inclement weather, frost, flood, capture) provide the basis for these factors (Wingfield, 1988). In addition, for an event to be a TPF, the organism must be either qualitatively, or quantitatively unprepared for the event. For example, if a gold fish (*Carassius*) is acclimated to summer temperatures of around 27°C, it cannot survive temperatures of 7°C, but if it is acclimated to 17°C, it can survive at 7°C, and even down to near freezing (Fry, *et al.*, 1942). A temperature of 20°C may be fine for a temperate fish, or even to a tropical fish, as this temperature is within their normal range. However, even such a moderate temperature is fatal to the Antarctic icefish *Trematomus* (Somero and DeVries, 1967), which is adapted to much colder temperatures (average temperature of -1.1°C), and normally experiences only very minor changes in temperature (yearly variation of around 0.1°C). TPFs stimulate the expression of a *transitory emergency state* (TES or stress response) which has many characteristics which are similar regardless of the TPF. The TES and TPFs will be discussed in great detail in Chapter III.

Transitions Between Phenotypic Stages

A transitional phenotypic stage may be induced by some environmental cue, with no outward expression of this transitional phase. For instance in example II, a bird can become photorefractory, even while still tending young (Wingfield and Farner, 1979). Development of the stage usually requires stimulation of the cell cycle, leading to cellular growth and differentiation, such as in the recrudescence of gonads in seasonal breeding birds. This stimulation probably involves the action of hormones acting either via genomic receptors (such as those of the steroids and thyroid hormones) or by slow acting membrane receptors which lead to cascades of messenger resulting in stimulation of the cell cycle.

Timing of development of the stage, as well as the onset of mature capability (Figure 1.7) are modulated by supplementary, integrating and synchronizing cues. These cues can either act as accelerators, or inhibitors, speeding up, or slowing down the progression of the phenotypic substages, as well controlling the "path" among physiological states within each substage. Onset of processes characteristic of a phenotypic stage (i.e. phenotypic substages; see below), are regulated by environmental cues in the same categories, but presumably would act throughout different mechanisms. These processes are probably regulated by messengers acting via fast acting membrane receptors to regulate the function of cells, rather than the slow acting effects via stimulation the

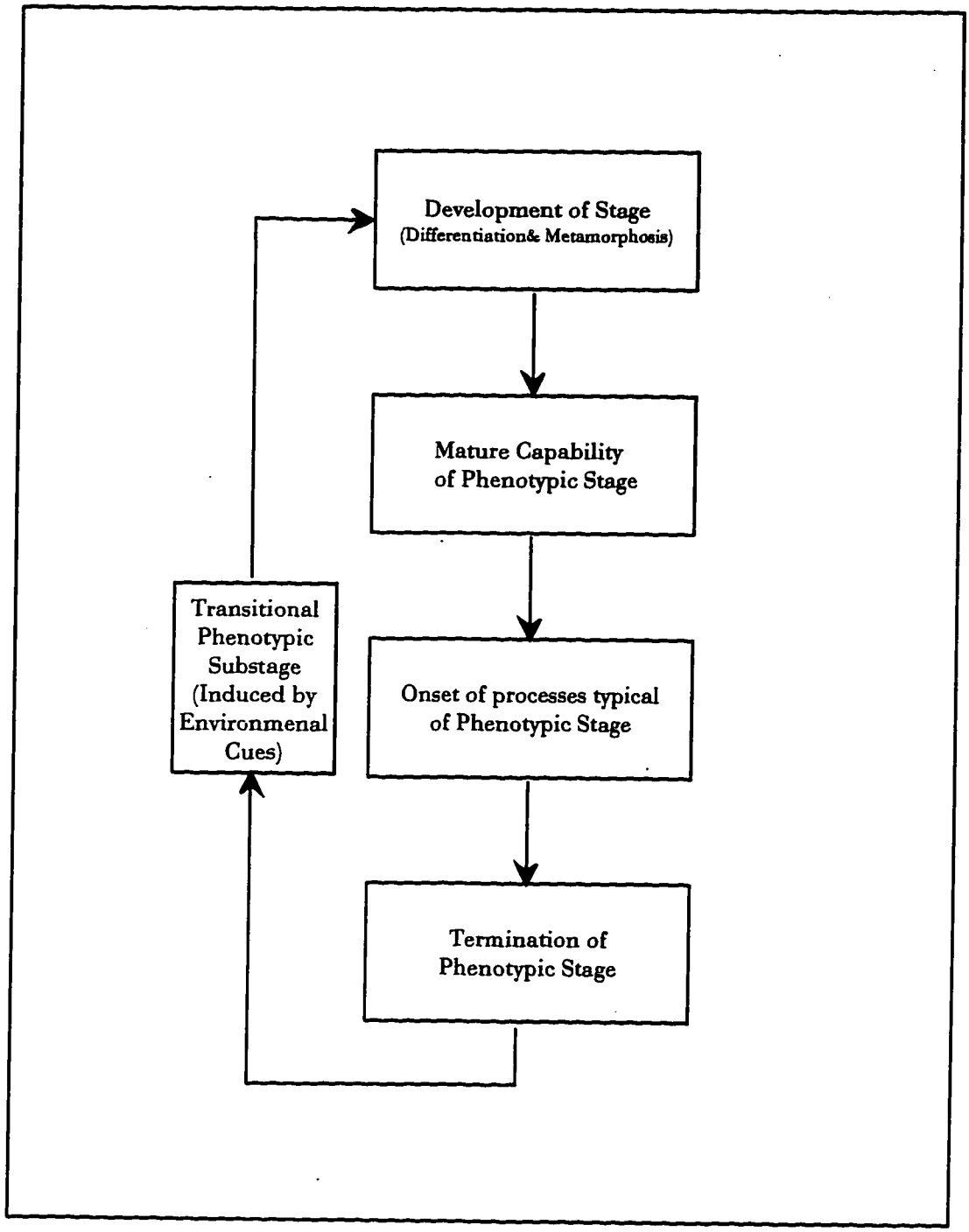


Figure 1.7 - Cyclic nature of the development and expression of phenotypic stages. Environmental cues (often initial predictive cues) trigger a transitional phenotypic substage that is committed to the next phenotypic stage. Development of the characteristics of the phenotypic stage may occur some conciderable time from the transitional phenotypic substage. Development of the structures involved in the phenotypic stage often occurs before the behaviors associated with the phenotypic stage.

cell cycle. Termination of a phenotypic stage may be initiated by any of the cues, including the unpredictable modifying factors.

A number of potential *physiological states exist* within each phenotypic stage (as well as within phenotypic substages characteristic of the stage). These physiological states represent the actual condition of the organism, including factors often included in the so-called extended phenotype (Dawkins, 1982), such as territory quality, and mate. The physiological state will change with time, associated with changes in the "extended phenotypic factors," as well as adjustments in the physiology of the organism associated with fluctuations in its environment. Note, these changes are triggered by environmental cues in a manner different from the transition between *phenotypic stages.*

If we look at the potential regulation of single genes, and the states of all cells within the individual, one sees an apparent infinite number of physiological states for an organism. But for the most part, a vast majority of these "micro-physiological states" will be functionally and selectively identical (or nearly so) and can be grouped into a finite number of *physiological states.* Although all the individuals within a physiological state will not be completely identical, their ability to interact with the environment will be indistinguishable.

Finite State Machine Model

If we step back and view an animal's life cycle at a coarse level, the life cycle appears as a series of discrete stages (i.e., reproduction, migration, molting...).

These *phenotypic stages* occur in a more or less cyclical progression, proceeding from one stage to the next in response to a wide number of environmental cues or stimuli. Within a given phenotypic stage, the organism can exist in a finite number of functionally different *physiological states.* The finite nature of these *phenotypic stages* and *physiological states* allows us to view the animal's life cycle as a finite state machine. These physiological states are similar to the concept of states as used by Mangel and Clark (1986) and Houston and McNamara (1992).

Finite state machines (FSM) are commonly used in computer programming literature to describe the decision making process of a program (see Gill, 1962). Three different type of variables are involved in a FSM: (1) excitation variables provide the stimulus for the machine, (2) response variables are the behavior of the system of interest to the investigator, and (3) intermediate variables, which are neither excitation or response variables, and are often difficult to measure as the occur within the machine. A FSM can be represented as a black box (figure 1.8). The "box" has a finite number of inputs (excitation variables), and finite number of outputs (response variables). The intermediate variables are inside the black box, and though they are important in determining the state of the machine, they are not easily determined, and can be ignored for dealing with transitions between stages. While intermediate variables can be ignored when dealing with state transitions, they are none the less quite interesting. In biological system, they encapsulate the physiological mechanisms within the organism and cells which

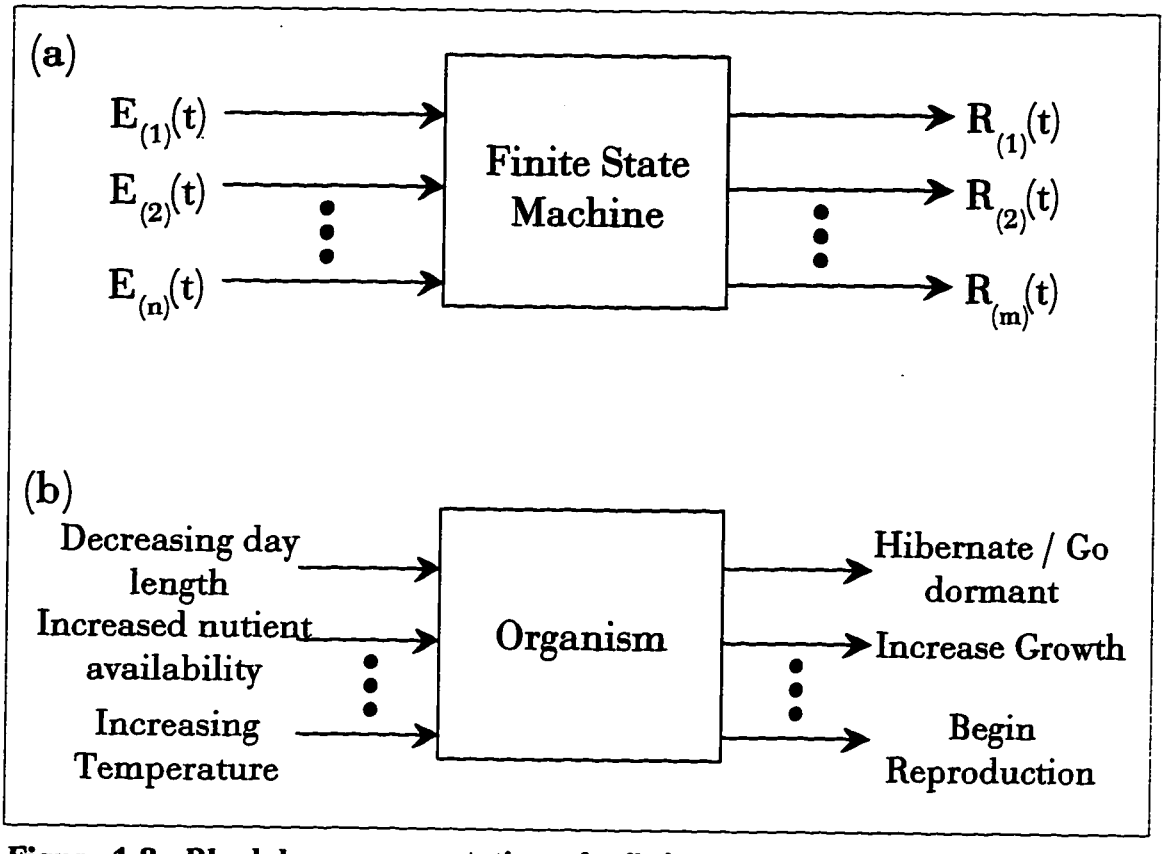


Figure 1.8 - Black box representation of a finite state machine. [Figure 1.1a redrawn from Gill, 1962.] (a) Inputs enter the black box from the left, and responses leave on the right of the black box. Excitation variables are represented as E, and response variables are represented as R, where t is the sampling time, and m and n are the number of response and excitation variables respectively. (b) Biological 'Black Box'. A vast number of excitations (stimuli) impinge on the organism. The animal has a finite number of responses to these excitations. Ellipsis (...) indicate that there are many different stimuli and responses for an organism, though only 3 are shown.

determine the actual state of the organism, and are thus very important, but their exact nature need not be understood to use the finite state model. However, a finite state model may indicate lines of investigation to determine mechanisms.

For a biological system, the excitation variables are the environmental cues described above (*e.g.* increasing day length, decreasing temperature, increasing food availability), and the response variables can be transitions between phenotypic stages and substages (*e.g.* begin gonadal recrudescence, initiate hibernation) or they may be changes in physiological state within a phenotypic stage or substage (*e.g.* increase growth, increase fat stores, abandon nest, and shiver to increase body heat) (Figure 1.8b).

Finite-state machines conform to several basic assumptions. The machine must be controlled by an independent *synchronizing source*. The system variables are not measured constantly, but rather at discrete intervals when certain specific events occur, called *synchronizing signals*. These discrete time instances are called the *sampling times*. An additional assumption of the system is that the behavior of the system at sampling time t_V (where v is an integer indicating the time interval) is independent of the interval between t_V and the previous interval t_{V-1}. Thus the system is not dependent on time, but rather on an ordinal number associated with the sampling times. This does not preclude events occurring at regular intervals, but such regularity is not required for the system. Environmental information can come from internal biological clocks as well. This independence from the interval

between samplings is important in biological systems, as they are responding to their environment, and not to the ticking of a simple clock. Time may be critically important, but only in that the cues from the environment will occur at more or less predictable times and the responses must coincide with appropriate conditions. Although biological clocks have been documented in a large number of biological systems (*e.g.* Aschoff, 1980, Gwinner, 1986, and Turek, 1985), these clocks are responsive to the environment, and the timing in the absence of environmental cues is usually different from timing in their presence. Thus, if a biological clock does not receive synchronizing signals from the environment, it loses its entrainment (Aschoff, 1980; Gwinner, 1986; Hastings *et al.*, 1989 and Turek, 1985).

Another basic assumption of the FSM is that the number of both the excitation and response variables is finite, and that the machine can only exist in one of a finite number of states during a given time interval. The current set of response variables (output) is uniquely determined by the present state of the machine, and the excitation variables at that time. The previous history of the machine is only important in that it uniquely determines the present state of the machine. The state at the next sampling time is solely determined by the present state, and the excitation variables. Thus, the machine can be characterized by a state function and a response function:

$$response_i = f_{response}(excitation_i, state_i)$$
$$state_{i+1} = f_{state}(excitation_i, state_i)$$

where $state_i$ is the state of the machine at *time* t_i (i = 1,2,3,....), $excitation_i$ is the set of excitation variables received during interval i, $state_i$ is the state of the machine at interval i, $state_{i+1}$ is the state of the machine in the next time interval after i, and $response_i$ is the response of the machine at interval i. Both the response function ($f_{response}$) and state function (f_{state}) have the current state of the machine ($state_i$) as a dependent variable, thus the response of the machine to a given excitation cannot be predicted, even if the response and state functions are known, unless the state to which the excitations are applied ($state_i$) is known, or the machine always has the same response to the excitation. Many excitations do not cause the transition to another state, but act to maintain the current state. For example, an excitation inhibits the transition to another state.

A simple example may better illustrate this state dependence. A coin toss game provides a very simple finite state machine (Figure 1.9). The coin has two sides, heads and tails. Points are only awarded for three heads in succession, without any intervening tails. In this game, the excitation variable are either heads or tails. The response variables are with score, or no score. There are three states: no heads in a row, one heads and two heads. The machine always responds to "tails" is in the same way (no score → no heads), and thus is always predictable, regardless of whether the state of the machine is known or not. Both

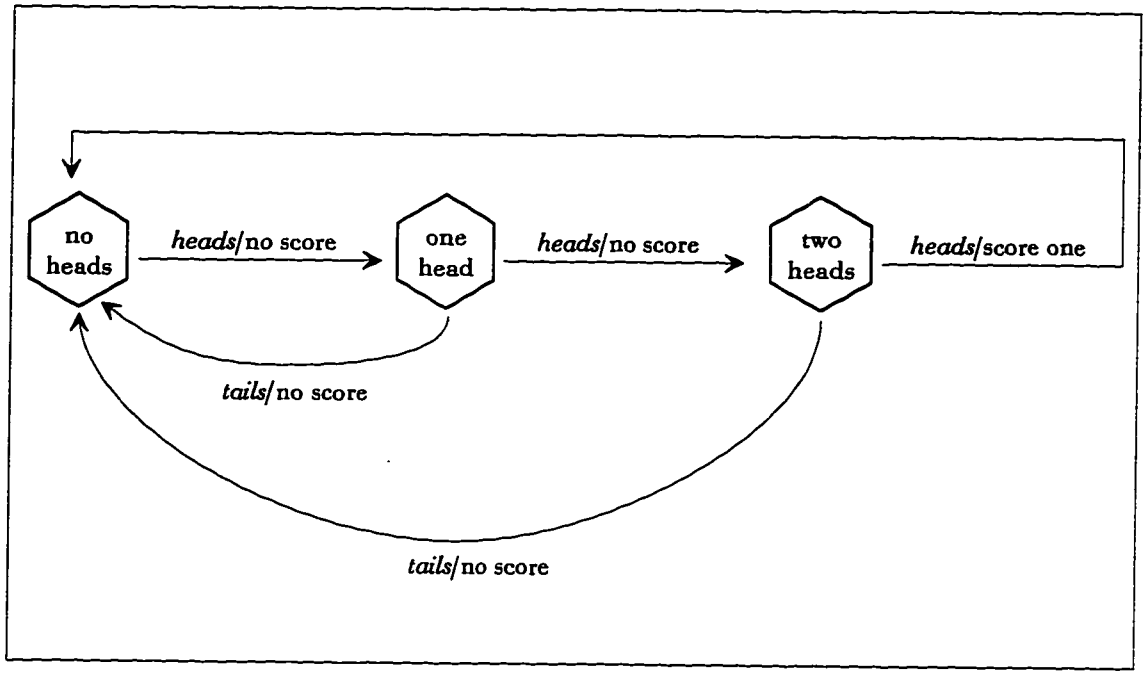


Figure 1.9 - Transition diagram of a simple coin toss game. In this game points are only awarded for three consecutive coin tosses resulting in heads. States are enclosed in hexagons, and excitations are in italics followed by the response (*excitation*/reponse).

the response, and the resulting state are ambiguous for "heads", unless you know the state of the machine. If you were to enter the room while the game was in progress, and the player flipped a "heads", you could not know whether the point was scored, without finding out from the players. You could not know whether the previous streak of heads was one, two, or zero, and thus could not predict the out come (response) even though you knew the excitation (heads).

Groups of states can be associated into a submachine which is in essence a "superstate" for the machine. Submachines can be composed of any number of layers, and can be used to encapsulate various layers of complexity (particularly when drawing transition diagrams.) When considering the transitions between submachines, transition between the states within the submachine can be considered as actions of intermediate variables.

Let's now examine a simple biological example of a finite state machine. Consider a stream animal, like *Hydra*, which is attached to the bottom, waiting for prey to be brought to it by the current. The prey are at a low density, so the *Hydra* is always hungry. For this example we will impose the limitation that while the Hydra is eating a food item, it cannot capture another prey item. As with all biological systems, the synchronizing source is the environment, and in this example the stream is the synchronizing source (as it moves the food to the animal). The environment changes through time, and this environmental variation provides the synchronizing signal. For our *Hydra* example the flowing

water is the synchronizing source. Environmental variation provides the excitation variables to the machine (the animal or plant), in the form of environmental cues (synchronizing signals). The machine integrates these cues, and responds appropriately to the state of the machine. For an animal, these states correspond to physiological/behavioral states.

In our simple *Hydra* feeding FSM, the excitation variables (see Table 1.1) are all supplementary cues: the presence or absence of a prey item and the completion of eating the food item. The Hydra has three responses: to wait, capture the prey, or to eat the prey. The "machine" has 2 states: eating, and waiting. During the eating state, the response set is: continue eating and wait. During the waiting state, the response set is: continue waiting, and capture prey. Even with only three different excitations in this model system, the reaction of the machine (response and transition to the next state) can not be determined, unless the state of the machine at the time of the excitation is known (see Table 1.2). This factor cannot be over emphasized, since it is important to all experimental manipulations of plants and animals: you can not predict the response of the system, unless you know both the physiological state of the organism, as well as the whole suite of excitations being experienced by the organism. A close examination of the transition diagram (figure 1.10) for this system shows this quite clearly. The response to the presence of prey is very different, depending on the current state of the *Hydra*.

Table 1.1 - **States, Excitation Variables, and Response Variables for a Simple Biological Finite State Machine.** A hypothetical freshwater stream predator remains attached to the bottom and captures prey from the stream as water flows past.

States
waiting
eating

Excitations
prey
no prey
prey devoured

Responses
wait
capture prey
continue eating

Table 1.2 - Transition Table for the Same simple Biological Finite State Machine as in Table 1.1. N.A. means that the excitation is not available to the current state of the machine.

	$response_v$			$state_{v+1}$		
$state_v$ \ $excitation_v$	prey devoured	prey	no prey	prey devoured	prey	no prey
waiting	N.A.	capture	wait	N.A.	eating	waiting
eating	wait	eat prey	eat prey	waiting	eating	eating

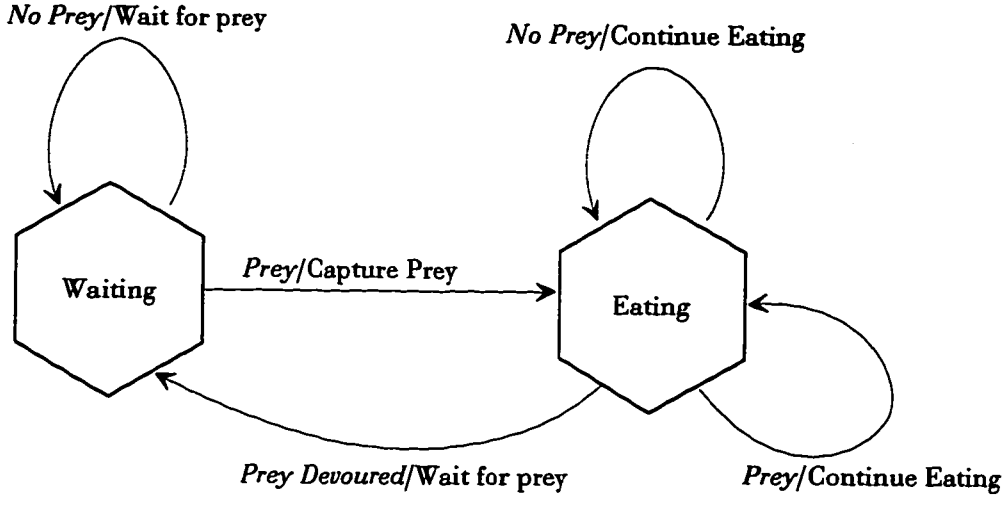


Figure 1.10 - Transition Diagram for a Simple Biological Finite State Machine. A hypothetical freshwater stream predator remains attached to a surface and captures prey from the stream as water flows past. The 2 states are represented by the hexagons, and excitations are in itallics. Responses of the machine in response to the excitation are to the right of the slash-marks for the corresponding excitation.

Application of Finite State Machine Models to Biological Systems

Example I. Modeling Sex Change in a Sequentially Hermaphroditic Fish

The bluestreak cleaner wrasse, *Labroides dimidiatus* , is a widely distributed coral-reef fish found from the Red Sea, down through the Great Barrier Reef of Australia (Meyers, 1991). It is a protogynous species, but unlike some other species of wrasses, all individuals are born as females (*i.e.*, no primary males). A terminal phase male (resulting from the sex change of a large female) guards a territory, and maintains a harem of 3-6 mature females, and several juvenile females (Robertson, 1972). When breeding, the male mates with each mature female once a day. A large number of aggressive interactions within the harem resulting in a dominance hierarchy within the harem. The male sits at the top of the hierarchy, keeping the dominant female (who is usually the largest) from sex reversing with aggressive threats and displays.

Death (or removal) of the territorial male initiates sex reversal in the dominant female within 1.5 hours. The reversal is first expressed as male-like aggressive and territorial behaviors, then male courtship and spawning behaviors develop in 2-4 days. By 14-18 days, the transformation is complete, and the terminal phase male is able to release sperm.

Sex reversal of females in this species appears to be inhibited by aggression of more dominant animals. Within the harem, more dominant females inhibit the reversal of lower status females by aggression. Aggression from the male inhibits

the reversal of the dominant female. Reversal is also dependent on the presence of a less dominant female, as an isolated female will not undergo sex reversal, probably because a lone female can join a harem and immediately begin breeding, but a lone male must find and establish its own territory, as well as attract females in order to breed.

In this system, the signals maintaining social status and sex come in the form of integrating cues from other members of the group (or the lack of those signals). Three phenotypic stages occur within the life history of this wrasse: (1) immature female, (2) mature female and (3) terminal phase male. The mature female phenotypic stage has three phenotypic substages: low-status female, unattached female, and dominant female. An individual may not go through all stages or substages, depending on their environment, and life span. Unlike many biological systems, the phenotypic stage transitions are not governed by initial predictive cues. The excitation set has 7 members, all but one of which are integrating cues: (1) reaching sexual maturity, (2) aggression from more dominant female, (3) no aggression from a dominant female + smaller females in harem, (4) no aggression from a dominant female + no smaller females in harem, (5) male aggression, (6) joining a harem with a larger female, and (7) joining a harem with only smaller females. These excitations do not come at specific times or at specific intervals. The response set is: (1) become low status female, (2) remain submissive, (3) become unattached female, (4) become dominant female, (5) remain dominant

female, (6) sex change to terminal phase male. These data can be used to create a transition diagram (Figure 1.11). The first thing that is noticeable from this diagram is that not all of the excitations are applicable to all stages, and that the response set for each state is more limited than the response set for the entire system (*e.g.* the terminal phase male cannot become a low status female). This property is characteristic of biological systems, because the present state of the system puts major constraints on the possible responses of the system, as well as the transition from one state to the next. The excitation set is composed of elements which represent a suite of stimuli (*e.g.* the combined effect of *No male aggression* + *No smaller females* leads to a different outcome from *No male aggression* + *Smaller females*). In fact, in biological systems the summation of cues often is critical to both the timing and the initialization of *phenotypic stage* transitions.

Example II. Modeling Phenotypic Stage Transitions of the Gambel's White Crown Sparrow

If we examine the life history of the migratory, temperate Gambel's White Crown Sparrow, *Zonotrichia leucophrys gambelii*, we see that the animal progresses between a number of discrete functional phases, such as migration, breeding, molting, and overwintering (Figure 1.12). These can be viewed as *phenotypic stages*, as they involve very different physiological expressions (in the broad sense as described above), and are hence phenotypically different with respect to

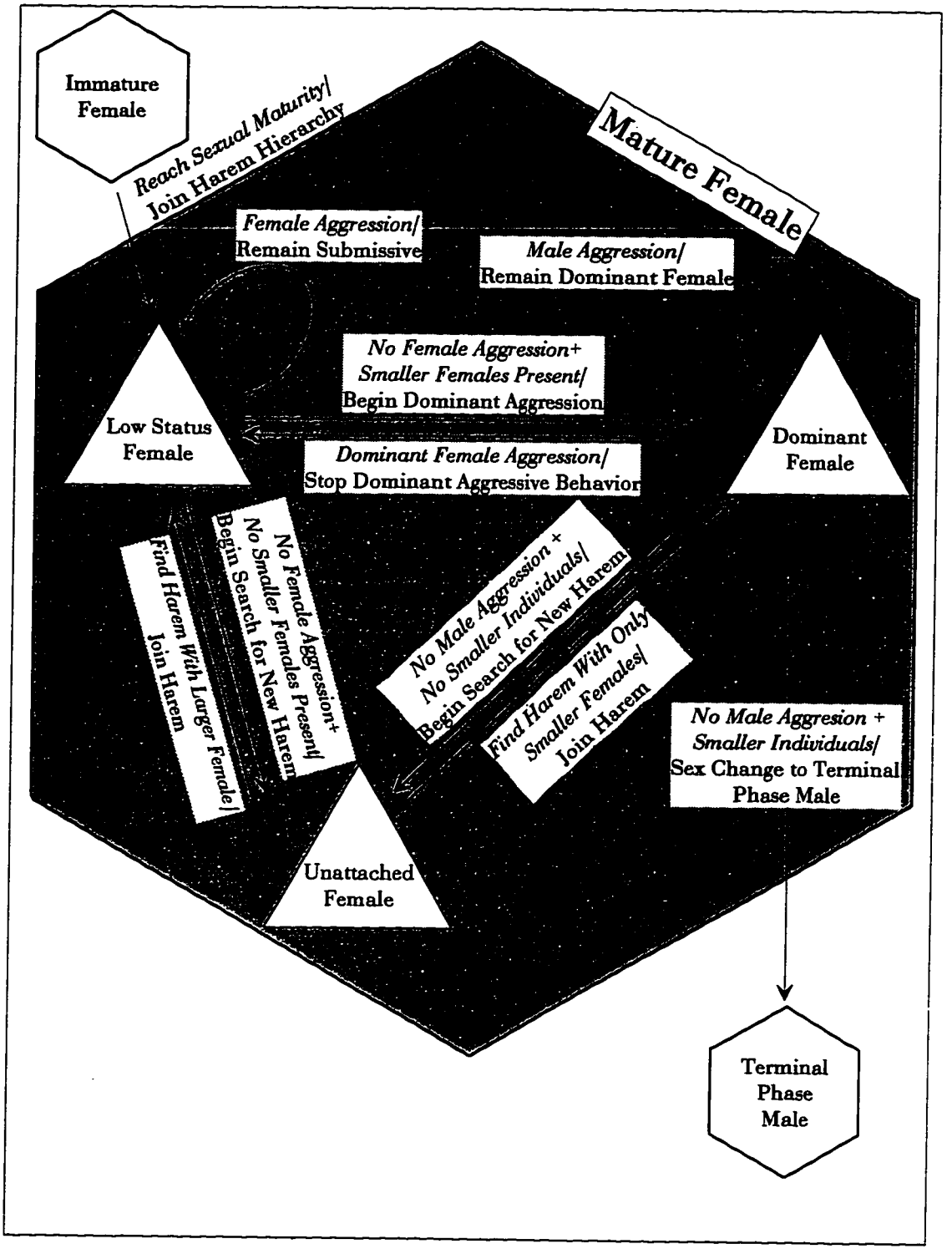


Figure 1.11 - Transition diagram for a sequentially hermaphroditic female Cleaner Wrasse *Labroides dimidiatus*. The diagram shows the transition between *phenotypic stages* in the life of a tropical reef wrasse. Phenotypic stages are within hexagons, and phenotypic substages are within triangles. The large mature female phenotypic stage includes three phenotypic substages.

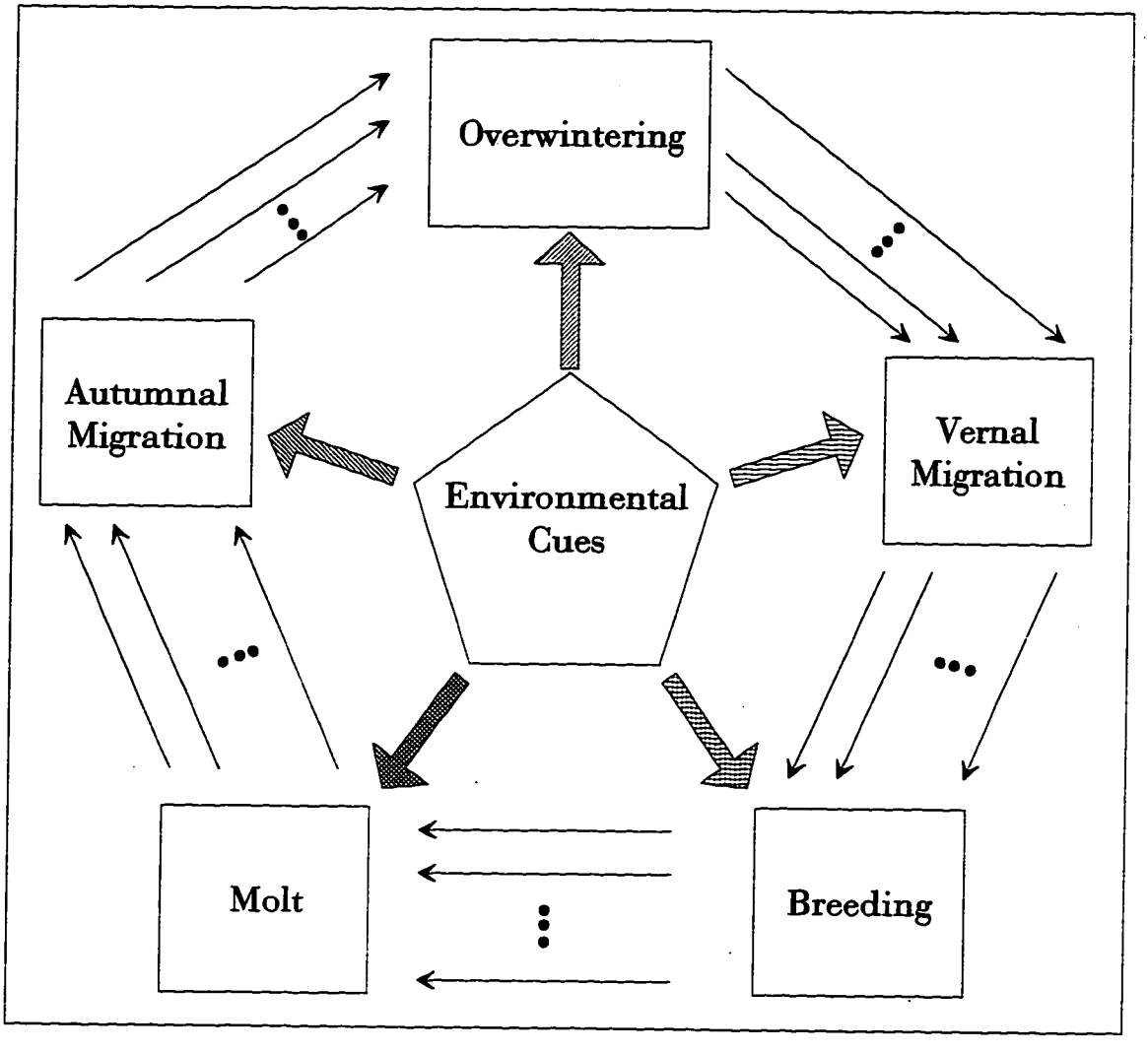


Figure 1.12 - The Seasonal Life Cycle of a migratory bird as diagrammed as a series of finite state submachines. Each submachine (drawn as boxes) receives a number of stimuli from the environment in the form of environmental cues (shown as large arrows). These stimuli are usually different for each submachine (as indicated by the different hatching of the arrows). The stimuli trigger a number of internal responses in the sub-machines (intermediate response variables), and also trigger the transition from one sub-machine to the next (indicated by simple arrows). The elipses (...) between the simple arrows indicate that there may be many ways to transition between submachines.

selection on the animal during the particular season. The transitions between the stages are controlled by cues from the environment, and these cues will obviously vary throughout the year (Figure 1.13). The transition between phenotypic stages can follow several different paths, and yield several different physiological stages, depending on both the integrated cues, and the initial physiological stage before the transition (Figure 1.14). Each of these *phenotypic stages* represents a finite state submachine, as each contains a distinct set of *phenotypic substages* (or super-states) designed for specific tasks. Within each of the *phenotypic substages* are a suite of *physiological states* that are determined by constraints imposed by both the environment and the genome. The exact nature of these *physiological states* will depend on such things as energy stores within the animal, quality of the territory, presence or absence of mate, and a whole set of physiological considerations as determined by the expression of the genome. Again, the excitation cues are important in determining the timing of the transition between *phenotypic stages* (submachines), but the state of the "machine"/individual is critical to the timing and direction of the transition. For a seasonally breeding bird, the transition from one *phenotypic stage* to the next is unidirectional, due the seasonal nature of the excitation cues (Figure 1.13). A bird that has migrated to the Arctic and begun breeding cannot then begin the vernal migration, because the state of the animal does not allow transitions to the vernal migration submachine (in part because the bird is already at the end point of migration), and the cues which normally trigger

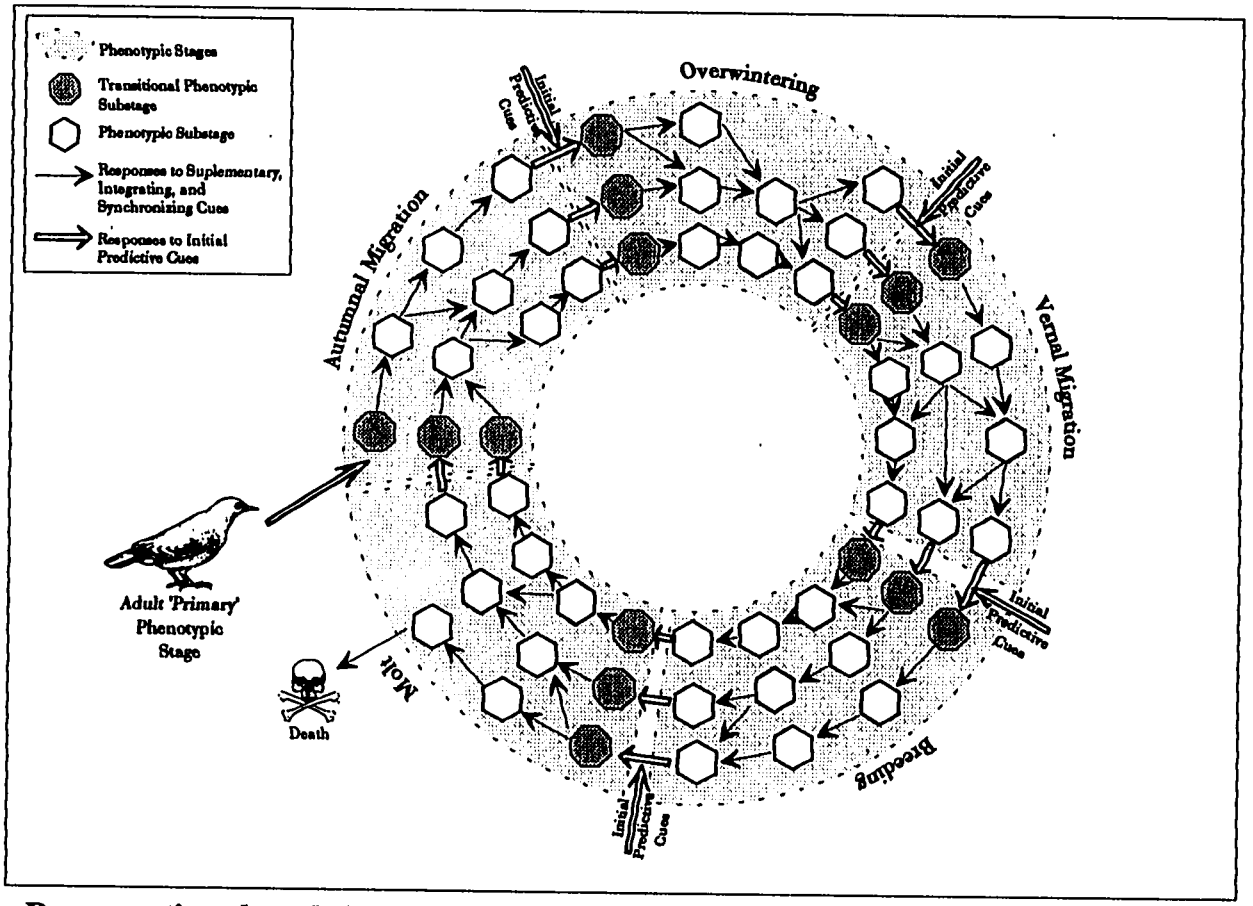


Figure 1.13 - Representing the adult life cycle of a bird as a finite state machine. The cycle is divided into a several phenotypic stages (breeding, vernal migration, autumnal migration, overwintering, etc.) which are finite state submachines acting to complete a specific task. Within each submachines there are a number of superstates, or phenotypic substages (see figure 1.14), dealing with a particular aspect of the phase. Each of the phenotypic stages typically has a transitional phenotypic substage which is usually triggered by initial predictive cues. Death, the terminal state for the life cycle, can occur at any time during the cycle, and is the only state which can be reached from all other states.

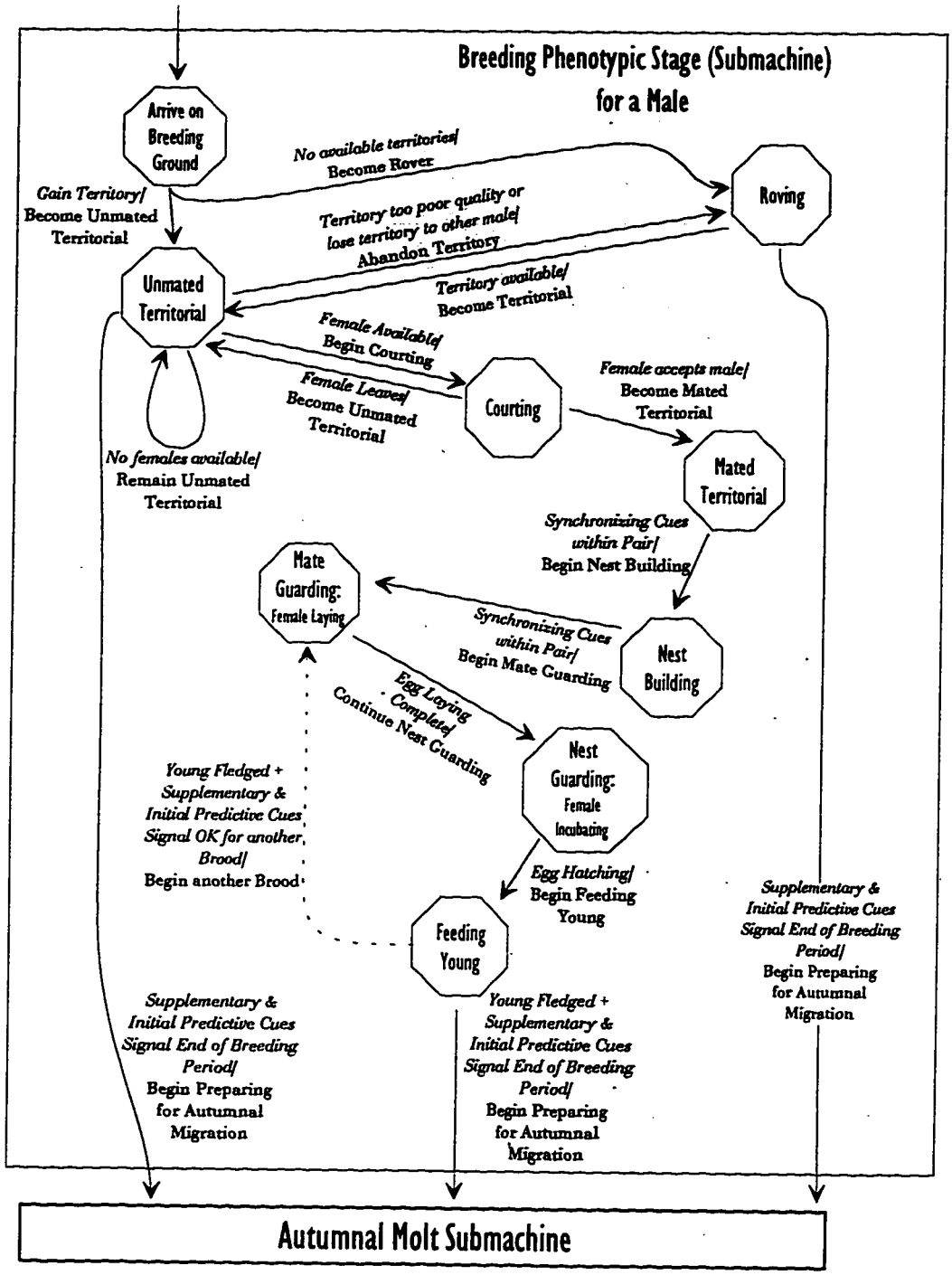


Figure 1.14 - Transition Diagram of the Breeding Phenotypic Stage for a male white crowned sparrow Zonotrichia leucophrys. Shows transitions between phenotypic substages (Finite State Submachines of the Breeding Submachine/*phenotypic stage*). This diagram depicts only the major transitions between substages, and does not attempt to show all transitions between *phenotypic substages*.

the vernal migration are not present during the late summer and early autumn, and also not present in the Arctic. As all transitions depend on the physiological state of the bird, there are several ways to transition between *phenotypic stages*, yielding different *physiological states* within the next *phenotypic stage.* The resulting physiological state depends on the initial state, and the combination of cues (excitations) leading to the phenotypic stage transition. Cycling of phenotypic substages can occur within a phenotypic stage, such as in the case of multiple broods (Figure 1.14), but after the bird has entered the next phenotypic stage, it must pass through the rest of the *phenotypic stages* before returning to the *phenotypic stage* it just completed (Figure 1.13). In cases where cycling of substages is possible (such as in the case of multiple brooded birds), the bird will integrate initial predictive and more proximal cues at the cyclic branch point to determine if another cycle is feasible. If, for example, the day length (an initial predictive cue) indicated that the breeding season was nearly over, the bird would not initiate another brood, even though the food supplies (a supplementary cue) might be very plentiful. The same would be true if the food supply was insufficient for another brood, the bird would not renest unless the initial predictive cues were correct. This cycling is most important in the breeding period.

For a seasonally breeding bird, such as the white crown sparrow, the synchronizing source will be the progression of the seasons as driven by the movement of the earth around the sun. At each step along the life cycle, the bird

will receive excitations in the form of cues from the environment. In some cases the cues will be important individually, but in other cases, the integration of several environmental cues combine to form a single excitation.

Hormones and the Regulation of Phenotypic Stage Transitions

How do animals integrate the signals from the environment, and bring about both stage transitions, as well as changes from substage to substage? Neural integration is important in the initial collation of the sensory inputs, but except for in purely behavioral responses, the signal must be sent to a large number of cells/tissues within the body. One might think that neural innervation of all the tissues would allow for immediate action on the integrated cues, but such a system would have several limitations. First the number of neural connections would be astronomical, and would probably require a massive enlargement of both the brain and spinal cord. Second, it requires that the animal maintain a large network of neurons that have no function for most of the year, as they would only be used during the transition periods, and only a small portion of these regulatory neurons would be needed for any one transition. Clearly these constraints on both the complexity of the system, and energy requirements to maintain such a network would be prohibitive.

The endocrine system may play one of its many roles in reducing the size and complexity of the regulatory systems of an animal. Hormones are carried throughout the body by the blood system and stimulate their target cells at very

minute concentrations. This non-directed transport of the hormones may at first seem a problem: because all cells are exposed to the hormone, how can specific cells and tissues be regulated? As all hormones require receptor proteins in/on their target cells to have their actions, the easy answer to this specificity problem is that the receptors are found on only certain cells, and only at specific times. The distribution of hormone receptors does not really solve our problem, as we then need to know how the distribution of the receptors was determined.

Cells respond to cues from their environment, just as do whole organisms. In many ways, cells within the body also function like finite state machines. First, they respond to excitations from the environment in the form of hormones, growth factors, neurotransmitters, cell-cell interactions, morphogens, and concentrations of nutrients/ions in the extracellular fluid. They also have specific responses to those excitations. The nature of the response will be determined by both the excitation and the state of the cell. A nerve cell that has a strongly hyperpolarized resting potential, will not generate a spike if exposed to a moderate amount of the proper neurotransmitter. If the resting potential of the same cell becomes depolarized to near threshold, an action potential will be generated in response to the same amount of neurotransmitter. If the neuron lacks receptors for a neurotransmitter, then that neurotransmitter ceases to be an excitation for the cell, even if depolarized to near threshold. The same is true for the action of hormones. If a cell lacks receptors for a hormone, the cell will not respond to the hormone.

Part of the state of a cell will be the presence or absence of hormone receptors on/in the cell. Also important to the state of the cells will be the effect of cues from previous time intervals. If the cell become terminally differentiated into a red blood cell or fused myotube, it is certainly not going to respond to any hormone or growth factor by beginning to proliferate. The state of the second messenger systems within the cell will also be critical to the cell's response to a hormone. If the cell normally responds to a hormone by elevating cyclic-AMP levels within the cell, and the cell has elevated phosphodiesterase in responses to other cues, then the hormone may have no effect on the response of the cell.

The actual response to the cell will also depend on the nature of the hormone receptors it has. The expression of hormone receptors will be regulated by the genome of the cell, by cues from the environment, and the state of the cell. During the development of the animal, many of the most important cues will be of the developmental class, as they will determine the location and final function of the cell within the animal. The number of receptors a cell has for a given hormone will vary over time in response to other hormones and concentrations of nutrients and ions within the extracellular fluids of the animal. Many hormones, such as the steroid and thyroid hormones, have permissive actions which set up the state within a cell, such that a second hormone can then have a specific effect that was not possible without the action of the permissive hormone. Also, many hormones have several types of receptors, which have different affinities to the

hormone, and are thus activated at different concentrations of the hormone. Cells with fewer receptors will require higher concentrations of hormones to respond.

This state dependence of cellular responses to hormones is crucial to regulating complex *phenotypic stage* transitions. This allows diverse cell types to respond to the same hormone in very different ways, and allows the same type of cells to respond to the same hormone differently at different times of year, depending on the state of the cell. A metamorphosing frog tadpole would be in serious trouble if all of its cells (rather than just those of the tail) began to die and be reabsorbed in response to the thyroxine signal. Also, this same thyroxine signal has no similar effect in adults.

This state dependent response helps to explain experiments on cellular responses to hormones that have often led to cellular responses that seem to be mutually exclusive, or inconsistent. Again, the response of the cell, even in culture, will be according to its state, so if the cells are in a different state, possibly by growing in media with differing concentrations of permissive hormones, the action of the hormone to be tested may be different. Local specificity of cellular responses to hormones will be determined by the state of the cell, as set-up by the developmental path of the cell, as well as the effect of other hormones. Thus a relatively small suite of hormones can serve to regulate very complex *phenotypic stage* transitions.

Conclusions and Implications

Several ideas develop from using a finite state machine model to understand responses to a changing environment. Because the progression of phenotypic stages is unidirectional, long-lived organisms will cycle through the stages (See Figure 1.12). The tight linkage of phenotypic stages to the environment means that a given phenotypic stage is only appropriate at specific season for the organism. Each of the phenotypic stages will have associated behaviors and other physiological adaptations, specific to that stage.

Within each phenotypic stage will be a number of phenotypic substages carrying out specific functions within the stage (See figure 1.15). A certain amount predictable of cycling between the substages (such as renesting for a second brood), or cycling may be due to unpredictable transient perturbation factors (TPFs), such as renesting after nest predation (TPFs will be covered in chapter III) can occur. The progression of substages follows a limited number of paths through the phenotypic stage and these paths are generally one directional, with a few predictable cyclic branches, and possible unpredictable set backs. In most cases, the phenotypic substages are found only within a single phenotypic stage, and thus restricted to specific parts of the life cycle.

During a phenotypic substage, an organism can exist in a number of physiological states (as defined above). The stage determines the organism's responses to the cues from the environment. From a given state, only a limited

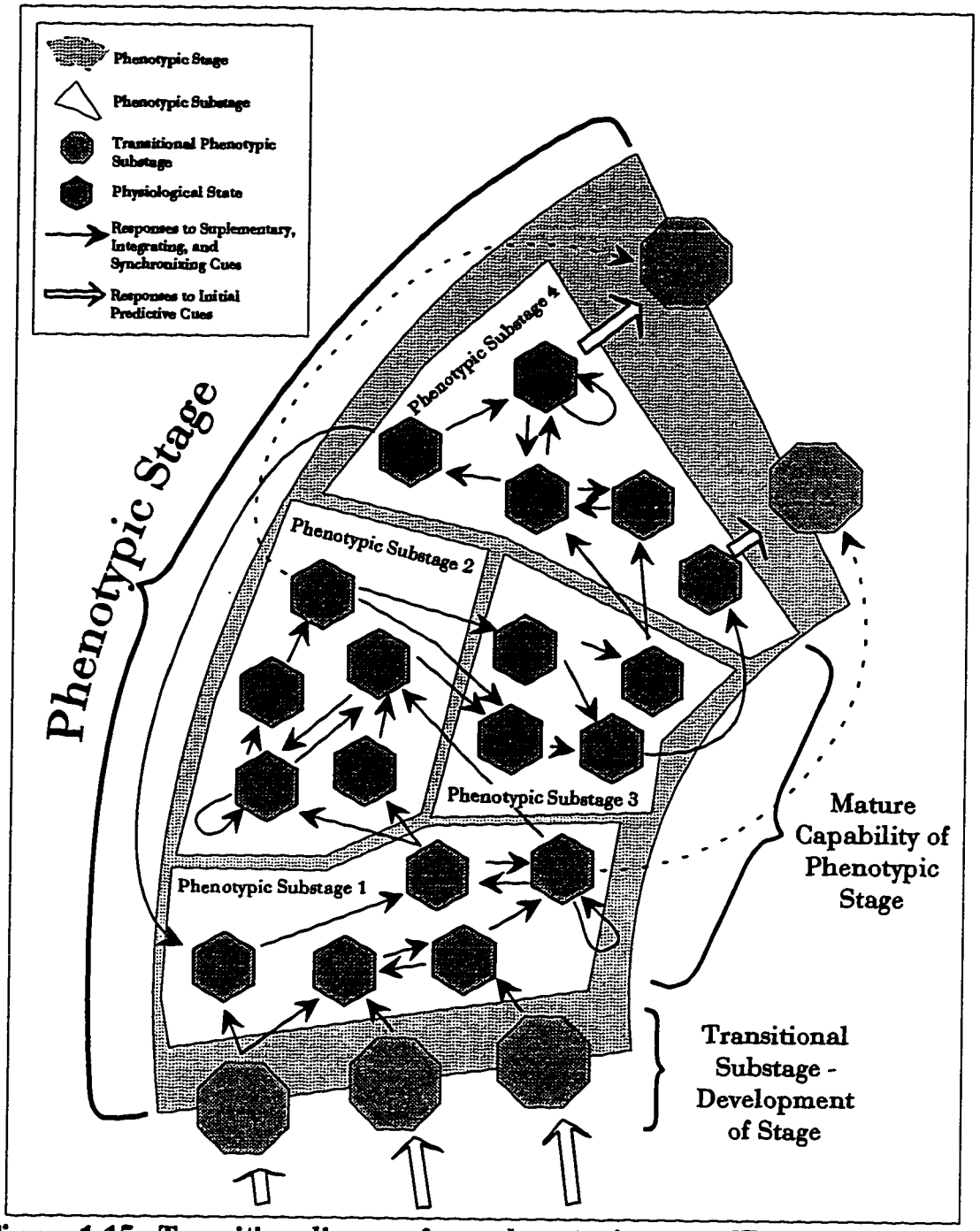


Figure 1.15 - Transition diagram for a phenotypic stage. The stage is triggered by initial predictive cues (though sometimes by supplementary and integrating cues), yielding a transition substage, where cellular processes (including possibly both differentiation and metamorphosis) are stimulated to produce the mature capability of the stage. Transition between substages and phenotypic stages are controlled by supplementary, synchronizing and integrating cues which combine with the initial predictive cues to provide the state specific excitations.

number of other states are obtainable in the next time interval. Such stage transitions are restricted to within a phenotypic stage, or into transitional substages, and are often restricted to within a single substage. Expression of physiological states, though generally progressing through the substage, is not one way, and there are many lateral detours, including those caused by TPFs. Physiological states are generally volatile and transition between states can occur quite rapidly. For example, a thirsty animal can quickly restore its water balance if given access to water. As the environment is changing constantly, so is the physiological state of the organism.

Environmental cues act on phenotypic stage transitions very differently to changes in physiological state. Because phenotypic stages are responding to a somewhat predictable environment, and the stage transitions generally involve stimulation of the cell-cycle, initial predictive cues are often most important for the regulation of these stage transitions. The precise timing of the stages is modulated by supplementary, synchronizing and integrating cues, as these cues can act to either accelerated or inhibit the progression of the stages. Transition between phenotypic states generally does not involve initial predictive cues, except at key points where the organism enters a transitional substage (See figure 1.15).

Hormones provide a possible mechanism for regulating these transitions. Transitions between phenotypic stages could be regulated late responses of hormones, or via genomic receptors, because stage transitions generally involve

activation of the cell cycle, differentiation, and gene expression. Expression of physiological stages is very rapid. While both early and late cell responses to the environment will occur, the transitions will result primarily from early cell responses via membrane receptors, including those for neurotransmitters.

This model provides a number of insights into understanding animals responses, but it also emphasizes a number of cautionary restrictions to experimenters. To make conclusions about the response of an organism to a stimulus (excitation), the experimenter must be able to determine the state of the organism. If overwintering animals are given an injection of testosterone, one can not expect them to respond in the same manner as the same animals during the breeding season. The experimenter also needs to know the normal excitations for the organism. Because excitations result from an integration of multiple cues, isolating a single cue and presenting it to the organism may result in responses not normally expressed in the field.

Thus the model proposed here provides a framework for investigating organismal responses to a changing environment. The model has features which are applicable to both whole organisms, and to cells, and provides insights into such broad questions as to why do animals have hormones.

Chapter II

Diversity Indices of Life History Stages and Their Sub-Stages: Implications for Endocrine Control Mechanisms.

Introduction

Ecologists have long been interested in describing the diversity of biological communities. It quickly became apparent, however, that simple species lists fail to account for the abundance of individual species. For example, a simple community may have 4 species each with equal abundance whereas another community may have 4 species in which one is common and the other three are rare. Do these communities have the same species diversity? Over three decades ago, information theory provided a basis by which to describe species diversity in terms of number of species and the abundance of each. Many methods have since been derived but one of the most commonly is the Shannon Index (MacArthur and MacArthur, 1961; MacArthur, 1964):

$$H = -\sum_i p_i \cdot \log_e p_i$$

where H = diversity index and p_i is the proportion of all individuals in the community that belong to species i. The Shannon Index will be highest if all species in a community have equal abundance, but will be lower if some species are more abundant than others.

Community ecologists have used this method and others not only to describe biological communities but also to try and determine what controls diversity. Investigations have related habitat complexity to variation in species diversity (MacArthur and Preer, 1962), as well as latitude (Ricklefs, 1987), competition and predation (Martin, 1988), disturbance (Connell, 1978) etc. These techniques have also been applied to other areas of biology. For example, the Shannon Index has been applied to describe and compare the length of birds' breeding seasons (MacArthur, 1964; Wyndham, 1986). In some avian species the breeding season is constant from year to year (*i.e.* only one or two months in which nesting occurs) whereas in others the breeding season may be highly variable covering 10 months in one year and only two months in another (see also Wingfield *et al.*, 1992, 1993). Thus, simply comparing the number of months that nesting can occur in any one year for each population may be misleading. Wyndham (1986) used the Shannon Index to calculate the number of equally good months (EGM) from the formula above where p_i is now the proportion of nests initiated in the i^{th} month. EGM will vary from 1, when all nest are initiated (eggs laid) in one month in all years to 12, when the number of nests initiated is equal in all months of the year in each year. Wyndham (1986) points out that EGM varies only as a function of the proportion of birds breeding in each month and does not differentiate between single and bimodal breeding seasons. Nevertheless, the calculation of EGM allows a much more precise comparison of breeding

seasons among species in relation to habitat, latitude etc. Application of these calculations to the breeding seasons of an Australasian species, the zebra finch, *Poephila guttata*, showed a highly seasonal component to breeding (Zann *et al.*, 1995). This species has long been considered a pure opportunist capable of breeding in any month in relation to unpredictable rains. However, the analysis suggests that although this species is capable of breeding at any time year there is still an underlying seasonal component. Clearly the control mechanisms for breeding in this species will be different than formerly supposed.

This approach also allows us to choose populations with similar or very different EGM values to then compare endocrine mechanisms experimentally. In this communication we have applied diversity indices (MacArthur, 1964) to life history stages and their sub-stages (see chapter one) within individuals of populations. We then go on to compare diversity of life history stages among populations and in relation latitude, habitat etc. Such comparisons will allow more meaningful assessment of endocrine mechanisms underlying transitions in life history stages and may shed further light on why there is such variation in hormone action among at least vertebrate taxa.

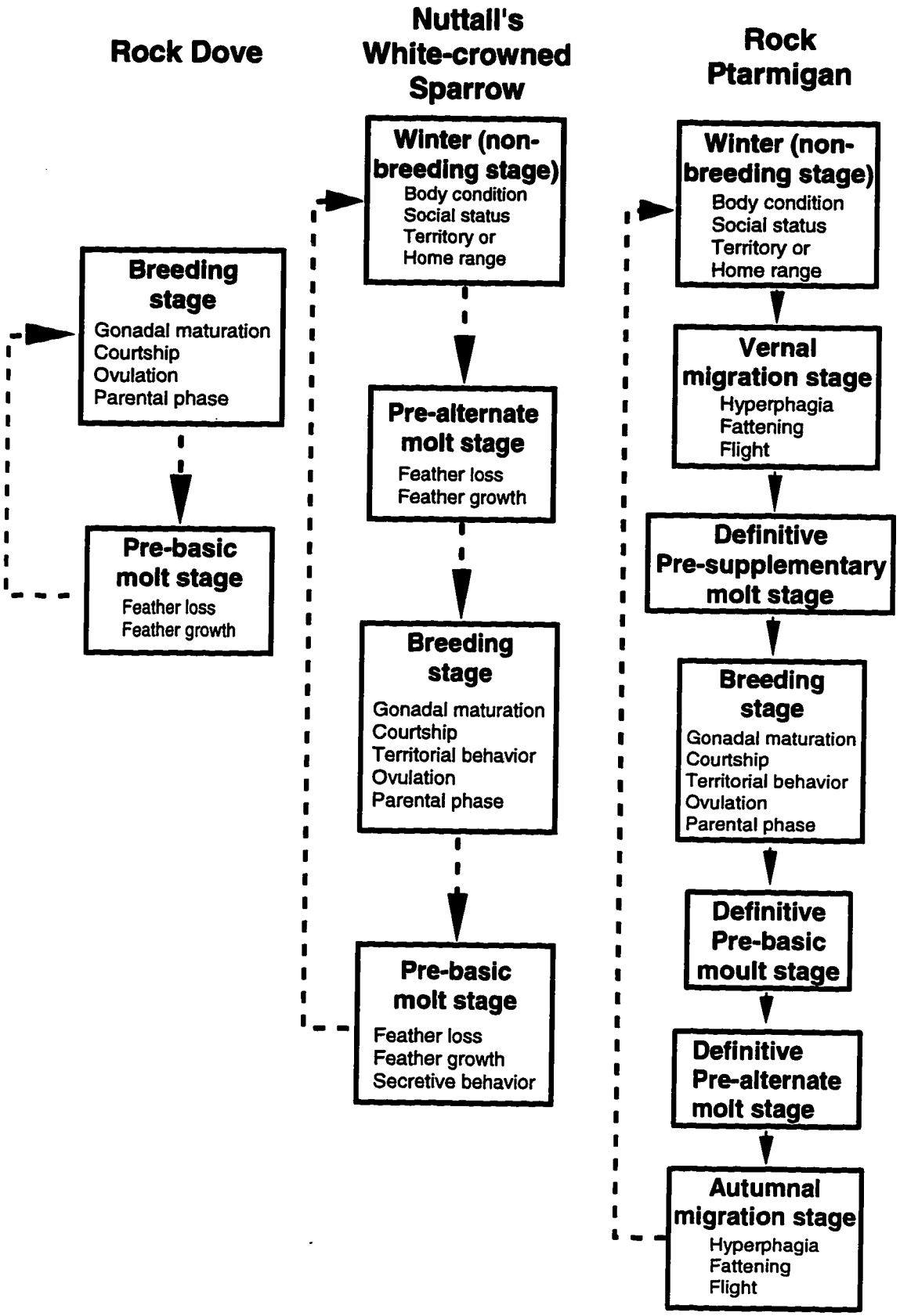
Finite State Machines and Diversity Indices.

A major role of hormones is to regulate transitions in morphology, physiological state and behavior. These transitions usually involve a period of development of a life history stage followed by a 'mature stage' in which a number

of sub-stages can be activated. Hormones may also play a major role in the activation of these sub-stages through mechanisms that are very different from their roles in promoting development (and also termination) of that stage (see chapter one). For each organism, then, we can postulate a finite state machine of several phenotypic (life history) stages that occur in a fixed sequence and often on a schedule determined by, for example, the changing seasons. Each life history stage is independent of the others although some may overlap to varying degrees, and each has a unique set of sub-stages (see chapter one). Some species may have more life history stages than others and even within an individual, some life history stages may have a more complex set of sub-stages than others.

The degree to which variation in life history stages exists is presented for three species of birds in Fig. 2.1. The rock dove (*Columba livia*) has only two obvious life history stages, breeding and molt (from Johnston, 1992), where as the Nuttall's white-crowned sparrow (*Zonotrichia leucophrys nuttallii*) has four (*e.g.* Blanchard, 1941; Mewaldt and King, 1977) and the rock ptarmigan (*Lagopus mutus*) has seven (Holder and Montgomerie, 1992). Note also that the number of sub-stages within each stage, and unique to that stage, varies. Comparing these finite state machines, it is clear that a problem remains as to how we compare the complexity of life history stages across taxa. Because these transitions of stage and activation of sub-stages are regulated by hormones, there are also likely to be major differences among species at cell and molecular levels. How do we compare

Figure 2.1 - Finite state machines of life history stages fot three avian species. In the rock dove (*Columba livia*) there are two major life history stages, breeding and molt, designated by the boxes. Dashed lines indicate the direction of progression from one stage to another. Note also that the number of sub-stages (finer print withn boxes) is greater for the breeding stage than for pre-basic moult. The center column represents four life history stages for the Nuttall's white-crowned sparrow (*Zonotrichia leucophrys nuttallii*). Again, dashed lines and arrows indicate the direction of progression. This temporal sequence cannot be reversed. Finer pront within boxes indicates the number of sub-stages. On the right is the most complex finite state machine with seven life history stages in the rock ptarmigan (*Lagopus mutus*). Dashed lines and arrows indicate the direction of progression, and finer print in boxes represents the sub-stages. Sub-stages in the three different moult stages here have been omitted but include feather loss and feather growth in each case. These may be different for each type of moult because different feathers may be moulted, and even if they are in the same feather tracts, the color of the feather that develops is different. So although the sub-stages in these different life history stages may appear to be the same, they may in fact have very different ecological bases and control mechanisms (see Wingfield et al., 1996 for discussion of other examples).


Rock Dove
Breeding stage
Gonadal maturation
Courtship
Ovulation
Parental phase
Pre-basic molt stage
Feather loss
Feather growth
Nuttall's White-crowned Sparrow
Winter (non-breeding stage)
Body condition
Social status
Territory or
Home range
Pre-alternate molt stage
Feather loss
Feather growth
Breeding stage
Gonadal maturation
Courtship
Territorial behavior
Ovulation
Parental phase
Pre-basic molt stage
Feather loss
Feather growth
Secretive behavior
Rock Ptarmigan
Winter (non-breeding stage)
Body condition
Social status
Territory or
Home range
Vernal migration stage
Hyperphagia
Fattening
Flight
Definitive Pre-supplementary molt stage
Breeding stage
Gonadal maturation
Courtship
Territorial behavior
Ovulation
Parental phase
Definitive Pre-basic moult stage
Definitive Pre-alternate molt stage
Autumnal migration stage
Hyperphagia
Fattening
Flight

life history stages across taxa, and what additional information does the finite-state machine give us? As with species diversity, simple number of stages (or species) may not be an accurate assessment. We can outline a number of questions that integrate all these possibilities and are derived from those questions posed by ecologists (*e.g.* Pielou, 1969, 1975):

a). Why are there not more (or fewer) life history stages and will their durations be the same? Another way of stating this would be, do the life history stages differ much in their tolerance ranges for environmental variables (see also chapter one).

b). Why are some stages widespread and others rare and are their relative proportions (durations) constant?

c). Are most, or all, of the life history stages within an individual fully adapted to the habitat at that time?

From Fig. 2.1 we see that some organisms have only a few life history stages and others have many. Thus we can apply formulae for ecological diversity (*e.g.* species diversity indices) to finite state machines to give us a measure of finite state diversity. This allows us to go beyond simply counting numbers of life history stages, and indicates the diversity of stages as a function of their duration as well as number.

Before finite state diversity can be measured, the temporal sequence of life history stages within an individual of a given population must be defined and

applied equally to all (see chapter one). This definition must specify the boundaries of the temporal sequence and how each stage is identified and its duration measured. The boundaries of the temporal sequence could be defined at many levels such as day, month, tidal fluctuation, wet season duration, year or longer). Numbers of life history stages must be determined by the same criteria in all populations. In chapter one I defined a life history stage as having a unique set of sub-stages (*i.e.* not shared by other life history stages), and each life history stage occurs in a set temporal sequence that cannot be reversed. For example in the white-crowned sparrow (Fig. 2.1), it is not possible to go back from the breeding stage to pre-alternate molt stage directly. It can only be attained by progressing through pre-basic molt and winter stages. On the other hand, the sub-stages within a life history stage can be activated in any sequence, or repeated many times within a life history stage (see below).

Duration of each life history stage can be measured directly from field observations. In the rock dove (Fig. 2.1), individuals are almost always in breeding condition once adult as indicated by developmental state of the gonads. Pre-basic molt, however, may be restricted to a few months of the year (Johnston, 1992). Thus, the durations of these two life history stages are very different but they also overlap almost completely. In the rock ptarmigan, a larger number of life history stages could mean shorter duration of each stage unless there is considerable overlap. In general, it is likely that some life history stages (such as winter and

molt) are more compatible in terms of overlap whereas others (such as migration and breeding) are much less compatible. The application of diversity indices is one way by which we can describe number, duration and overlap of life history stages across all populations of all taxa.

Finite State Diversity

Bearing in mind the admonition of Pielou (1969; 1975) that we should be careful in how we apply mathematical techniques to biological problems and not take too many liberties in stretching assumptions made in mathematics to highly complex biological processes, we have applied the Shannon diversity index (formula given above) to finite-state machines of life history stages. We use the ideas of MacArthur (1964) to use this formula to calculate the diversity of life history stages as a function of their number, duration and degree of overlap. In this way we are able to formally describe finite state machines of different populations as a frame work to formulating hypotheses about hormonal control. We feel that this objective method will allow us to identify which populations should be compared to give maximum power in determining those mechanisms.

To illustrate these points we present some theoretical examples first as a way of defining our methods. The boundary of the temporal sequence we define here as one year with intervals of months (*i.e.* 12 intervals in Fig. 2.2). The vertical axis is then made up of different life history stages (for example 6 stages in Fig. 2.2). For a given temporal sequence of hypothetical life history stages we can

MSID = 2, TSD = 6

	Interval											
Stage	1	2	3	4	5	6	7	8	9	10	11	12
1		●										
2			●	●								
3					●	●						
4							●	●				
5									●	●		
6											●	●

MSID = 4, TSD = 3

	Interval											
Stage	1	2	3	4	5	6	7	8	9	10	11	12
1	●	●	●	●								
2			●	●	●	●						
3					●	●	●	●				
4							●	●	●	●		
5									●	●	●	●
6	●	●									●	●

Figure 2.2 - Schematic representation of finite state diversity. Each column represents an interval in a temporal sequence of hypothetical life history stages. In this example the intervals are months in a year. The rows represent hypothetical life history stages; in this example we have six. The hatched circles indicate in which interval each life history stage is expressed by an organism in a given population. In the top panel, each stage has a duration of 2 months and there is no overlap. In this case, the mean stage interval diversity (MSID) is calculated as 2 and the temporal stage diversity (TSD) is 6. In the lower panel, the duration of each state is now four months so that at each interval, two stages are expressed simultaneously. MSID now equals 4 and the TSD 3. See text for details.

use the formula above to calculate the mean stage interval diversity (MSID) which is a measure of the diversity in duration of each life history stage; and the temporal stage diversity (TSD) which is a measure of the diversity of life history stages as a function of not only their duration, but also number and degree of overlap. Both of these measures may have considerable import for hormone mechanisms (see below). Some hypothetical examples will now show how MSID and TSD vary.

If we assume six life history stages each of two months duration (indicated by the shaded circles corresponding to each stage and interval in Fig. 2.2 upper panel) and with no overlap, then this gives us a MSID of 2 (*i.e.* all are the same - two months duration) and a TSD of 6 (there are 6 life history stages with no overlap). Note also that in these finite state machines, the organisms must be in at least one life history state at all times (otherwise it obviously would not be alive). If we now assume six life history stages each with four months duration (lower panel of Fig. 2.2), then we get a MSID of 4 (*i.e.* all stages have the same duration of 4 months) and a TSD of 3 (there is 50% overlap of all life history stages with the organisms expressing two in each interval.

We can go on to more complex hypothetical examples. The top panel of Fig. 2.3 is the same as that for Fig. 2.2 (for comparison, six life history stages each with two months duration). If we then assume six life history stages with unequal duration (one to four months, Fig. 2.3 middle panel) then we get a MSID of 2.7

MSID = 2
TSD = 6

Stage	Interval 1	2	3	4	5	6	7	8	9	10	11	12
1	●	●										
2			●	●								
3					●	●						
4							●	●				
5									●	●		
6											●	●

MSID = 2.7
TSD = 4.5

Stage	Interval 1	2	3	4	5	6	7	8	9	10	11	12
1	●	●										
2			●	●	●							
3						●						
4							●					
5								●	●	●	●	
6												●

MSID = 4.5
TSD = 2.7

Stage	Interval 1	2	3	4	5	6	7	8	9	10	11	12
1	●	●	●	●	●	●	●					
2								●				
3									●			
4										●		
5											●	
6												●

Figure 2.3 - Schematic representation of finite state diversity. Each column represents an interval in a temporal sequence of hypothetical life history stages. In this example the intervals are months in a year. The rows represent hypothetical life history stages; in this example we have six. The hatched circles indicate in which interval each life history stage is expressed by an organism in a given population. In the top panel, each stage has a duration of 2 months and there is no overlap (identical to the top panel of Fig. 2.2 for comparisons). In the middle panel, the duration of each stage is variable but there is no overlap. Here MSID=2.7 and TSD=4.5. In the lower panel we have an extreme hypothetical example where one stage has a duration of seven months and the rest have a duration of one month each. MSID is now 4.5 and TSD is 2.7. See text for details.

and a TSD of 4.5. In the lower panel of Fig. 2.3 we have six life history stages with an extremely skewed duration. One stage covers seven months (intervals) and the other 5 are of one month (interval) each with no overlap. In this case the MSID = 4.5 and TSD = 2.7. Clearly the diversity of life history stages is not simply a function of how many there are, but also functions of duration of each stage and the degree of overlap with other stages. Is this variation in finite state diversity reflected in natural examples?

If we now go to our avian examples from Fig. 2.1 and apply the diversity indices, once again we see considerable variation (Fig. 2.4). In the rock dove the breeding stage is found in up to 10 months and the molt in six months. This gives a MSID of 8.5 (both are relatively long in duration) and a TSD of 1.3 (very few stages and with considerable overlap). In the white-crowned sparrow (mid panel of Fig. 2.4), the four life history stages are of shorter duration with no overlap giving a MSID of 3.8 and a TSD of 3.1. The rock ptarmigan (lower panel of Fig. 2.4) has seven life history stages with varying degrees of overlap. In this case, MSID = 3.4 and TSD = 3.4. From these data it now becomes clear that although the application of the Shannon index is useful for comparing finite state diversity across taxa, it has a drawback in that in some cases very different temporal sequences of life history stages can give similar MSID and TSD values. Compare the number and temporal sequence for the white-crowned sparrow and rock ptarmigan in Fig. 2.4, they appear very different but the MSID and TSD values are

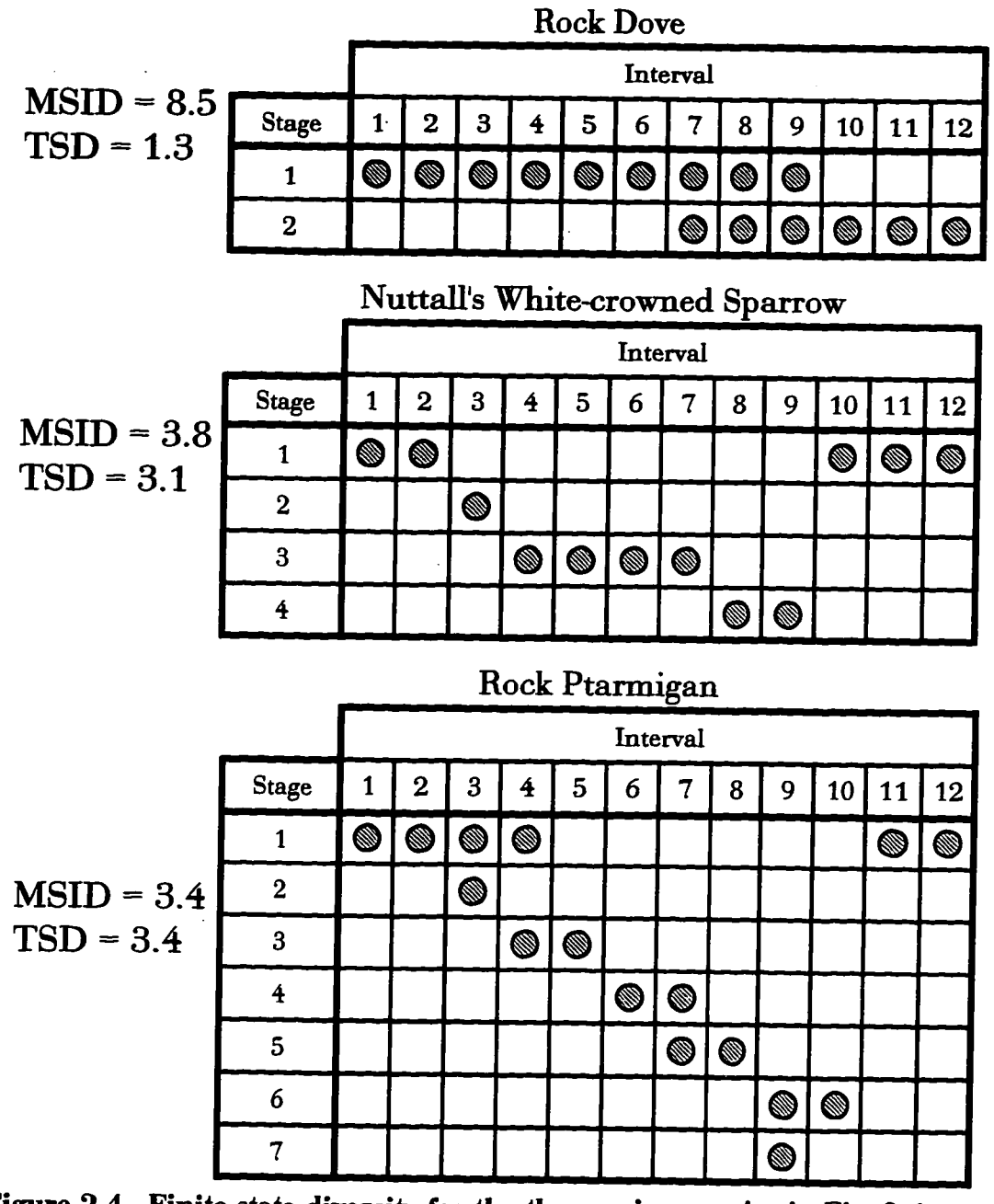


Figure 2.4 - Finite state diversity for the three avian species in Fig. 2.1. Each column represents an interval in a temporal sequence of hypothetical life history stages (i.e. months of the year). The rows represent life history stages that vary according to species. The hatched circles indicate in which interval each life history stage is expressed by an individual in a given population. In the rock dove (top panel) there are two life history stages with relatively long durations and considerable overlap. MSID is high and TSD low. In the white-crowned sparrow and rock ptarmigan, there are more life history stages with shorter durations and varying degrees of overlab. Here the MSID ad TSD values for ech axon are similar. See text for details.

very similar. Exactly the same problems have been encountered by ecologists when using this formula, and others, to determine species diversity (Pielou, 1969, 1975). For these reasons community ecologists have largely abandoned this approach. However, in this case, we feel that this property of the method may indicate useful information. Here we use the technique to describe the finite state diversity not define it. Thus is it possible that although the number and temporal sequence of life history stages in rock ptarmigan and white-crowned sparrows appear to be different superficially, the MSID and TSD values suggest they are very similar. Does this mean that the hormonal mechanisms underlying control of each stage are also similar? Conversely we could argue that since the finite state diversity of these species are very different from those of the rock dove, then the latter species may have very different control mechanisms. These hypotheses generated from the diversity index are eminently testable and will allow us to quickly identify extremes within taxa and similarities across widely diverse taxa. We can then test whether mechanisms differ or show convergence accordingly. This has the potential to provide a basic framework for environmental endocrinology.

Before going we should also point out that we are aware that the Shannon index should be ideally applied to an infinite population (or collection of stages and intervals, Pielou, 1969, 1975). This is obviously not the case for finite state machines with finite numbers of life history stages and intervals in which they are

likely to occur. We also calculated the Brillouin index for a finite population (Pielou, 1969, 1975) and compared the MSID values using the two indices for finite state machines from several species and populations of species (Fig. 2.5). Clearly the MSID values using the Shannon Index and Brillouin index in relation to the number of phenotypic stages (*i.e.* life history stages) are superimposable. Because the Brillouin index poses program problems owing to factorials within the formula, we use the Shannon index throughout.

Finite State Diversity in Different Taxa

Natural histories for several species and populations within species were used to determine the number of phenotypic stages (life history stages) and their estimated duration in months. From these data the MSID and TSD were calculated and plotted against the number of phenotypic stages (Fig. 2.6). Species from which data were compiled are named in the legend to Fig. 2.6 with references. MSID decreases as the number of phenotypic stages increases (Fig. 2.6. upper panel). This is expected as the duration of each stage must go down as the number of stages increases. However, this relationship is not a simple straight line. There can be considerable variation in MSID for species with similar phenotypic stages (*e.g.* stages 2 and 3 in the upper panel of Fig. 2.6). Similarly, there can be wide variation in phenotypic stages for a particular MSID (*e.g.* see the range of stages for the MSID value of 3.5-4.5 in Fig. 2.6).

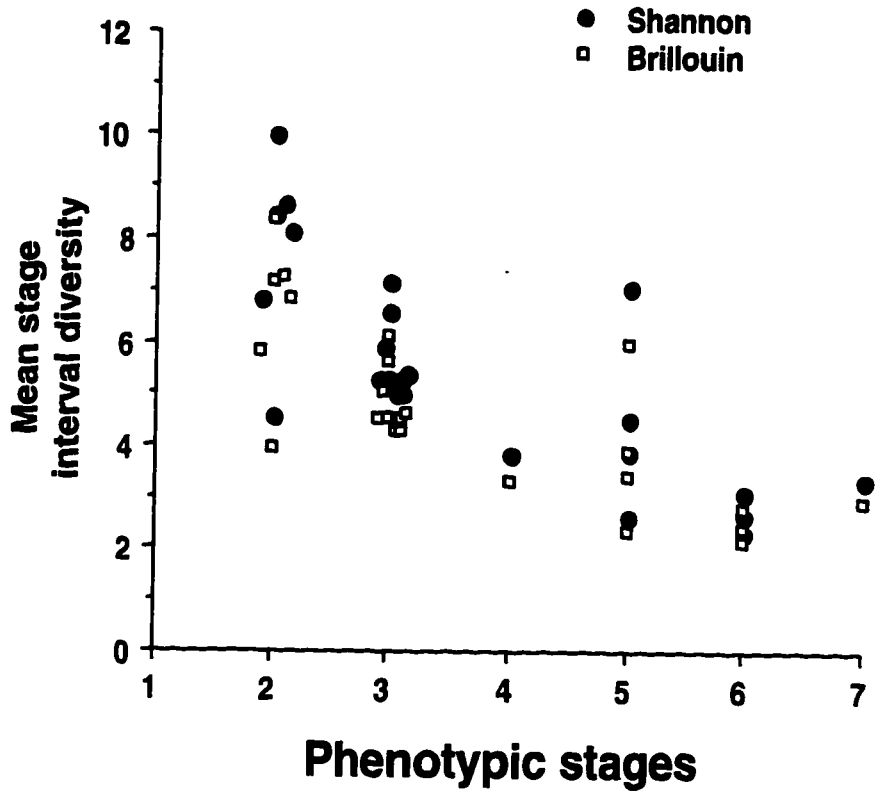


Figure 2.5 - A comparison of mean stage interval diversity calculated by the Shannon or Brillouin indices in relation to life history stages (phenotypic stages) in various avian taxa and populations. See legend for Fig. 2.6 for list of species. Values from the two methods are superimposable.

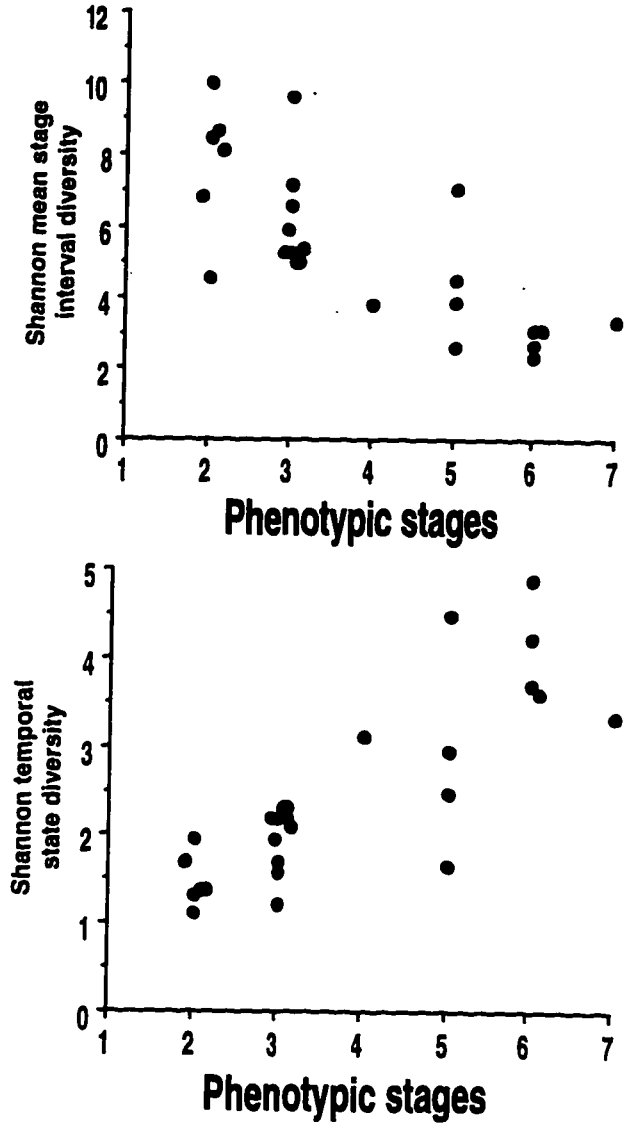


Figure 2.6 - Mean stage interval diversity (upper panel) and temporal stage diversity (lower panel), in relation to phenotypic stages (life history stages) of several taxa and sub-populations. These diversities were calculated from the Shannon index. Data were compiled from the following species: masked booby, *Sula dactylatra*, (Anderson, 1992); Smith's longspur, *Calcarius pictus*, (Briskie, 1992); European starling, *Sturnus vulgaris*, (Cabe, 1992). sooty tern, *Sterna fuscata*, (Chapin, 1954); semipalmated sandpiper, *Calidris pusilla*, (Gratto-Trevor, 1992); rock ptarmigan, *Lagopus mutus*, (Holder and Montgomerie, 1992); rock dove, *Columba livia*, (Johnston, 1992); house sparrow, *Passer domesticus*, (Lowther and Cink, 1992); Cassin's auklet, *Ptychoramphus aleutica*, (Manuwal and Thoresen, 1992); rufous collared sparrow, *Zonotrichia capensis*, (Miller, 1962); Inca dove, *Scardafella inca*, (Mueller, 1992); Laysan albatross, *Diomedea immutabilis*, (Whittow, 1992); white-browed sparrow weaver, *Plocepasser mahali*, (Wingfield et al., 1991); western gull, *Larus occidentalis wymani*, (Wingfield et al., 1982); zebra finch, *Poephila guttata*, (Zann et al., 1995); red crossbill, *Loxia curvirostra*, (Hahn, 1993); snow bunting, *Plectrophenax nivalis* (J.C. Wingfield and L.M. Romero unpublished); and several populations of white-crowned sparrow, *Zonotrichia leucophrys*, and song sparrow, *Melospiza melodia*, (Blanchard, 1941; Blanchard and Erickson, 1949; Mewaldt and King, 1977); Wingfield and Farner, 1978a,b; Wingfield and Hahn, 1994).

For TSD the trend is opposite to MSID as we would expect (lower panel of Fig. 2.6). As the number of phenotypic stages increases then TSD should also increase but again this is not a perfect straight line. There is wide variation in TSD for species with identical numbers of phenotypic stages (*e.g.* stage 5 in Fig. 2.6 lower panel). Additionally, TSD may be similar over a range of phenotypic stages (*e.g.* TSD value of 1.5-2.5 in lower panel of Fig. 2.6). These data suggest that calculating TSD and MSID an indicate much about variation in finite state diversity for different species. Are the mechanisms underlying control of the temporal sequence of phenotypic stages the same for MSID values of 10 and 4 when only two phenotypic stages are expressed (*e.g.* Fig. 2.6 upper panel) or for TSD values of 4.5 and 1.5 when 5 phenotypic stages are expressed? Conversely, are the mechanisms underlying the regulation of transitions from stage to stage similar when MSID values are similar across several species with different numbers of phenotypic stages? These kinds of analyses can indicate which comparisons can provide the most critical experiments to determine how individuals orchestrate morphological, physiological and behavioral adjustments throughout their life cycles.

It is also possible to conduct these kinds of analysis at different taxonomic levels. Comparisons of closely related species within a sub-family, *e.g.* the emberizinae (Fig. 2.7) show that the relationships of MSID, TSD and numbers of phenotypic stages are more linear. This is even more striking if a single genus is

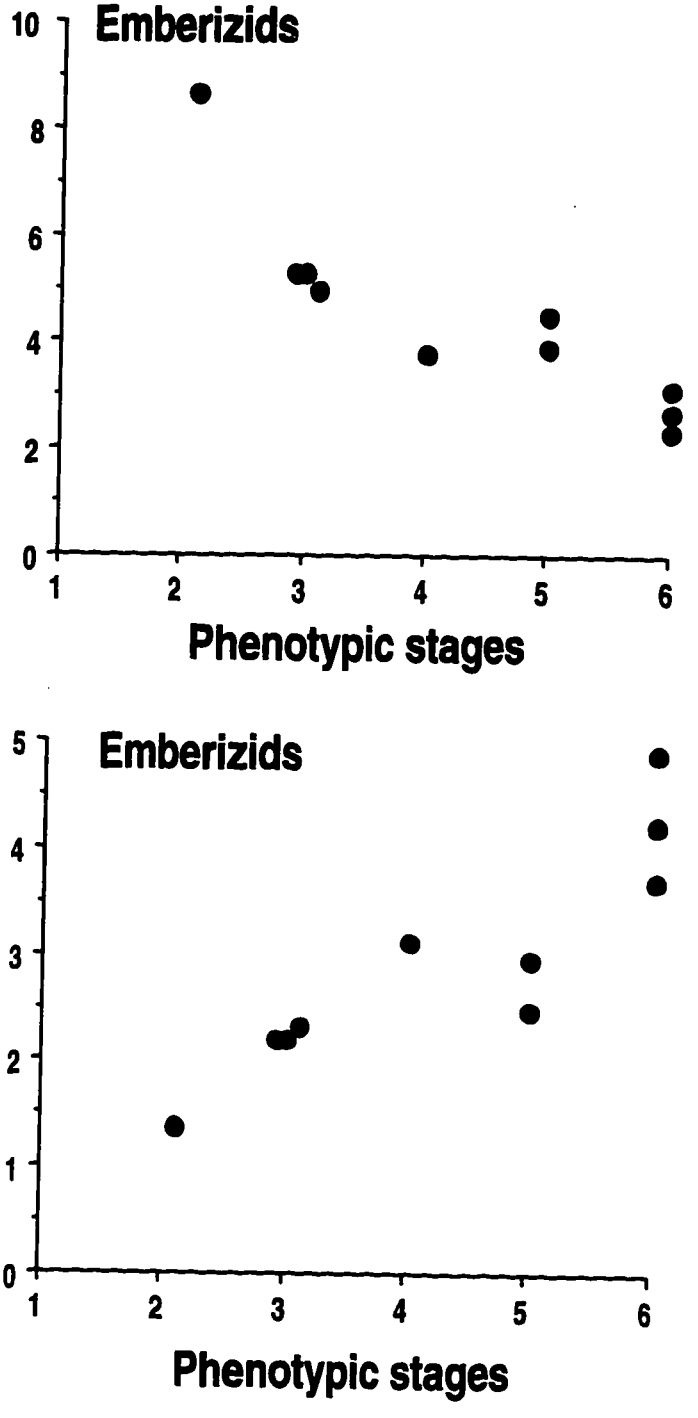


Figure 2.7 - Mean stage interval diversity (upper panel) and temporal stage diversity (lower panel), in relation to phenotypic stages (life history stages) of several taxa and sub-populations. Mean stage interval diversity and temporal stage diversity in relation to the number of phenotypic stages in a sub-family of passerines (Emberizinae). See legend for Fig. 2.6 for species and references.

compared (*e.g. Zonotrichia*, Fig. 2.8). Here then it is possible to conduct experiments on congeners to determine how mechanisms differ along a defined array of finite state diversities. Going one step further to look at populations of a single taxon of song sparrow (*Melospiza melodia morphna*), we can see that there is little change in MSID or TSD even though this one population has representatives with two different phenotypic stages (Fig. 2.9). Does this indicate that the mechanisms underlying transition from one stage to another are identical within these sub-populations? Clearly these methods allow us to analyze finite state diversity in a way that provides clear hypotheses and identifies which taxa will provide the best comparisons - even among very closely related taxa.

Assessing the Complexity of Sub-stages Within Each Phenotypic Stage.

Each phenotypic stage has a unique set of sub-stages (Fig. 2.1). Whereas phenotypic stages are expressed in a fixed temporal sequence, the sub-stages can be expressed in variable sequences or sets. From Fig. 2.1 it is clear that the number of sub-stages within each phenotypic stage can vary greatly from species to species and from stage to stage within an individual. Since the activation of these stages requires hormonal and neural input, probably in response to appropriate environmental cues, then the control mechanisms may vary from species to species within a type of stage as well as among different stages within an organism. Thus, there is a need to formally analyze the degree of complexity of sub-stages within

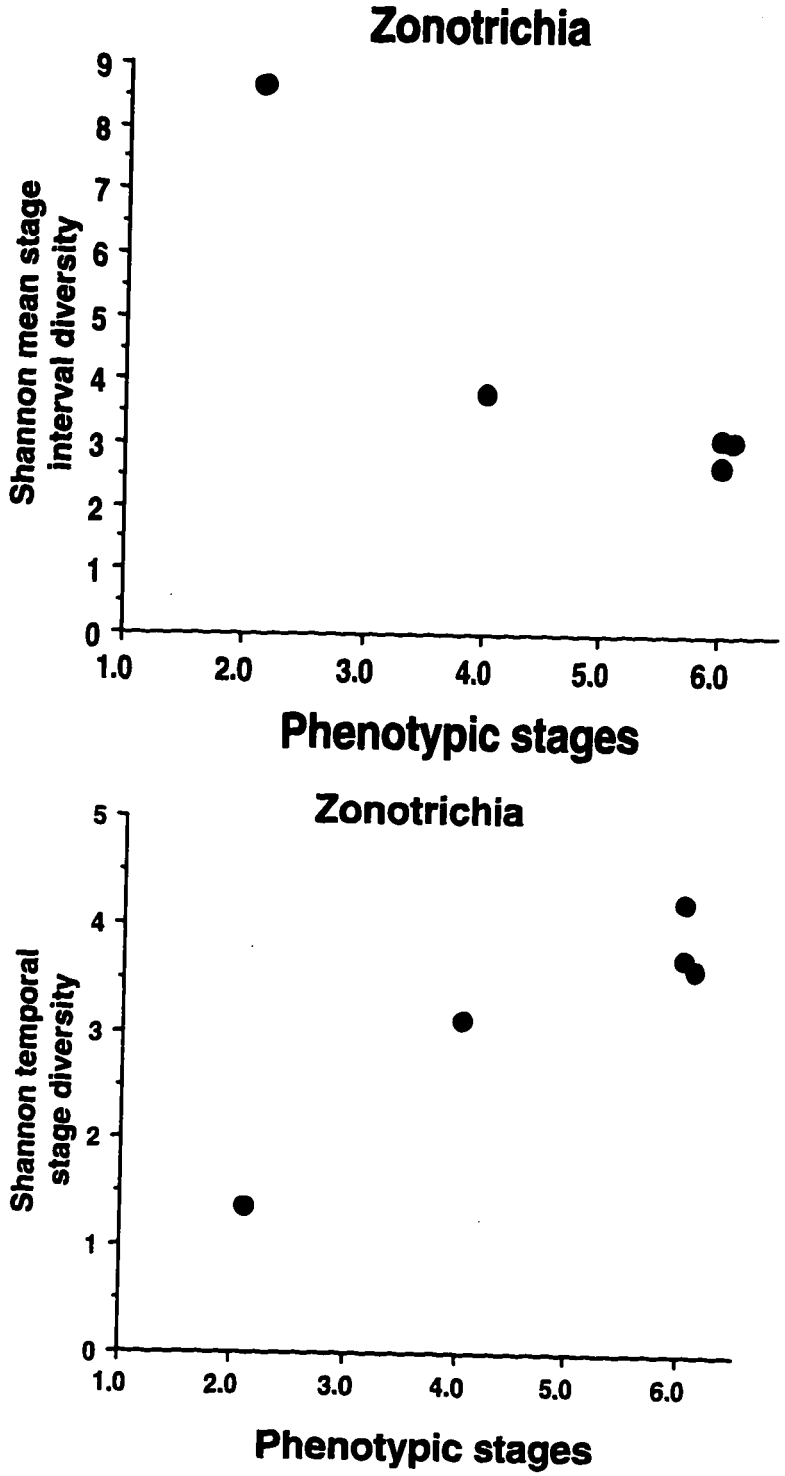


Figure 2.8 - Mean stage interval diversity and temporal stage diversity in relation to the number of phenotypic stages within a single genus, Zonotrichia. Taxa are: *Z. capensis*; *Z. leucophrys gambelii*; *Z.l. nuttallii*; *Z.l. pugetensis*, and *Z.l. oriantha*. See legend for Fig. 2.6 for references.

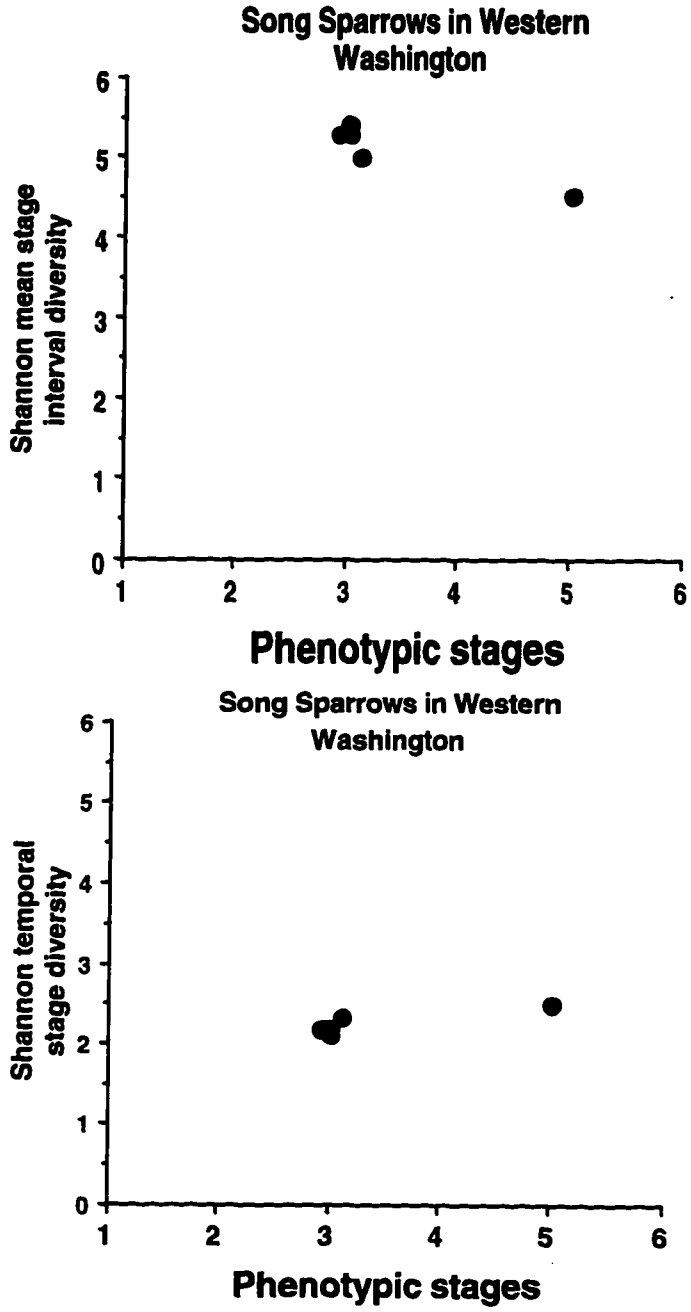


Figure 2.9 - Mean stage interval diversity and temporal stage diversity in relation to the number of phenotypic stages within a population of song sparrow, *Melospiza melodia morphna*, in western Washington State. These sub populations were studied at a sites on the Pacific Coast, lower Puget Sound, San Juan Islands, Mid and high elevations of the Cascade Mountains. All were at similar latitude (47-48° N). One of these populations (mountain) has two extra phenotypic stages - spring and autumn migrations between the lowlands and mountain passes (from Wingfield and Hahn, 1994; J.C. Wingfield unpublished).

each phenotypic stage, not only the number of sub-stages but also how each is connected to other sub-stages.

An illustration of variation in complexity of sub-stages is given in Fig. 2.10 for the migratory white-crowned sparrow, *Zonotrichia leucophrys gambelii.* The breeding stage (Fig. 2.10a) is complex and consists of at least eight distinct sub-stages. Firstly gonad development is necessary and then several sub-stages can follow such as establishment of a breeding territory, courtship and pair-bonding, nest building, and ovulation/copulation/egg-laying. Following this sexual phase then several parental sub stages can be expressed such as incubation, feeding nestlings and fledging. There is a temporal sequence in broad terms but note that if the nest or eggs are lost to predator then the various sexual sub-stages can be expressed again. Also if the population is multiple brooded then sexual and parental sub-stages can be repeated. This does not mean, however, that all sub-stages are equally connected with other sub-stages. In Fig. 2.10a we show the main known connections of sub-stages. The maximum number of connections is calculated from the formula **n(n-1)**. In Fig. 2.10a the maximum number of connections possible is 56 but only 19 (ratio of 0.34) have been identified in field studies of white-crowned sparrows (from Blanchard, 1941; Blamchard and Erickson, 1949; Mewaldt and King, 1977; Wingfield and Farner, 1978a,b).

The migration stage is less complex (Fig. 2.10b) consisting of a migratory readiness (zugdisposition) followed by periods of hyperphagia and fattening and a

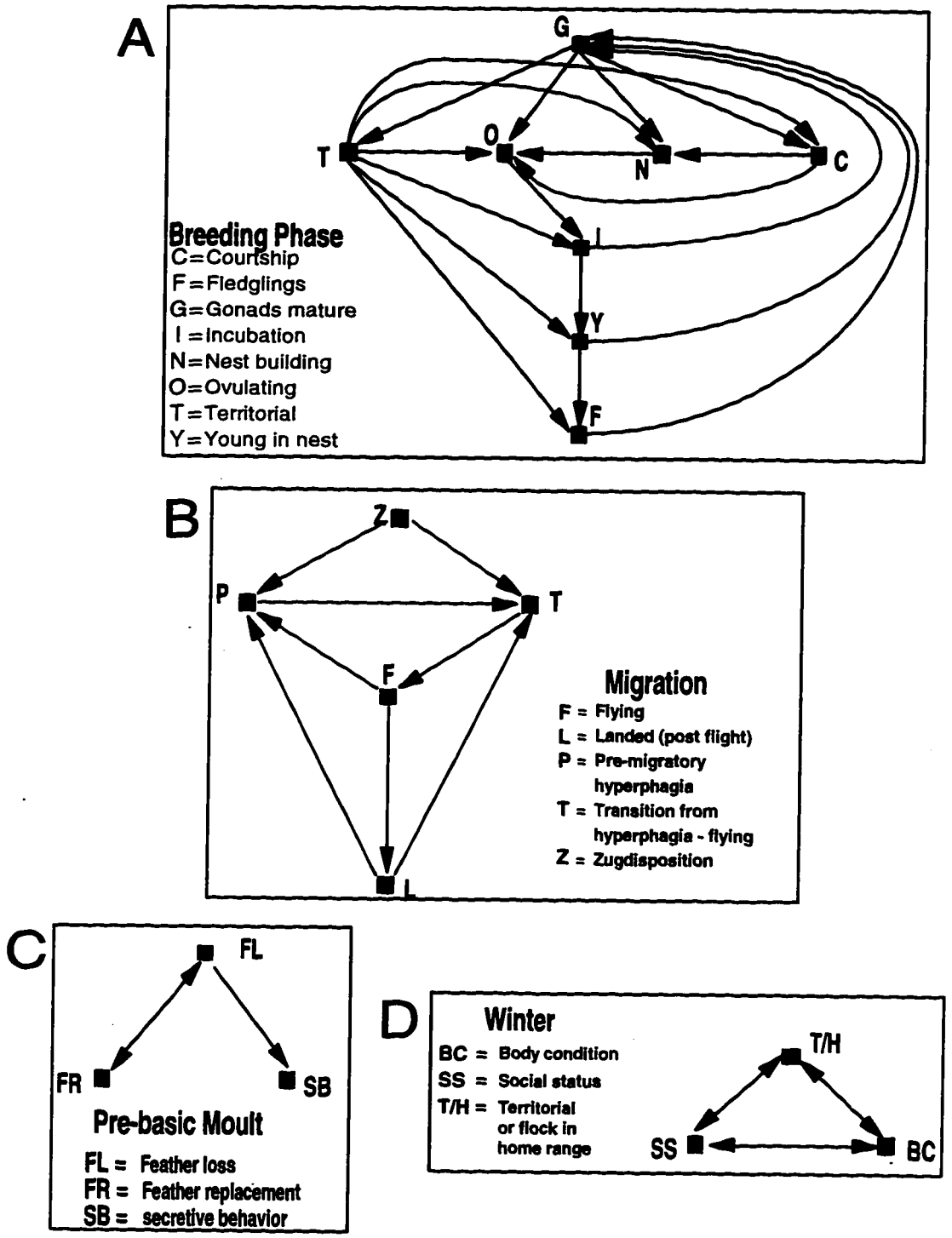


Figure 2.10 - Schematic representation of connectivity of sub-stages within life history stages of the migratory white-crowned sparrow, *Zonotrichia leucophrys gambelii*. Note that not all interconnections of sub-stages may actually be possible. Also, note that the diversity of sub-stages varies markedly with life history stage even within a population.

transition period before an actual flight. After landing there is another transition back to hyperphagia and fattening (see Wingfield *et al.*, 1990). Here the maximum number of connections is 20, but only 8 (ratio of 0.4) are demonstrated (Fig. 2.10b). For the pre-basic molt stage (Fig. 2.10c) there are only three sub-stages and the ratio of connections is 0.5. For the winter stage there is also a smaller number of sub-stages but connectivity is maximal (*i.e.* 1.0. Fig. 2.10d). Clearly connectivity varies not only as a function of the number of sub-stages but also as a function of known relationships and sequences of events within a phenotypic stage.

We can also compare the sub-stages within the same phenotypic stage across different taxa. For example, the connections of sub-stages in the breeding stage are different depending upon whether the species is single brooded (Fig. 2.11a) or double brooded (Fig. 2.11b). The potential number of connections is 56 in each case, but the number of actual connections is only 16 (ratio of 0.29) for single brooded and 0.34 for multiple brooded. For a brood parasite (*e.g.* the brown-headed cowbird, *Molothrus ater*, Dufty and Wingfield, 1986a,b) there is no parental phase at all because the females lays her eggs in the nests of other species. Here there are only 4 sub-stages (Fig. 2.11c) with a connectivity of 0.5. One can also compare individuals within a population but with different breeding stages (Fig. 2.12). Breeding females that show parental care will have a connectivity ratio of about 0.34 as in Figs. 2.10 and 11 (Fig. 2.12a). However, many males show no

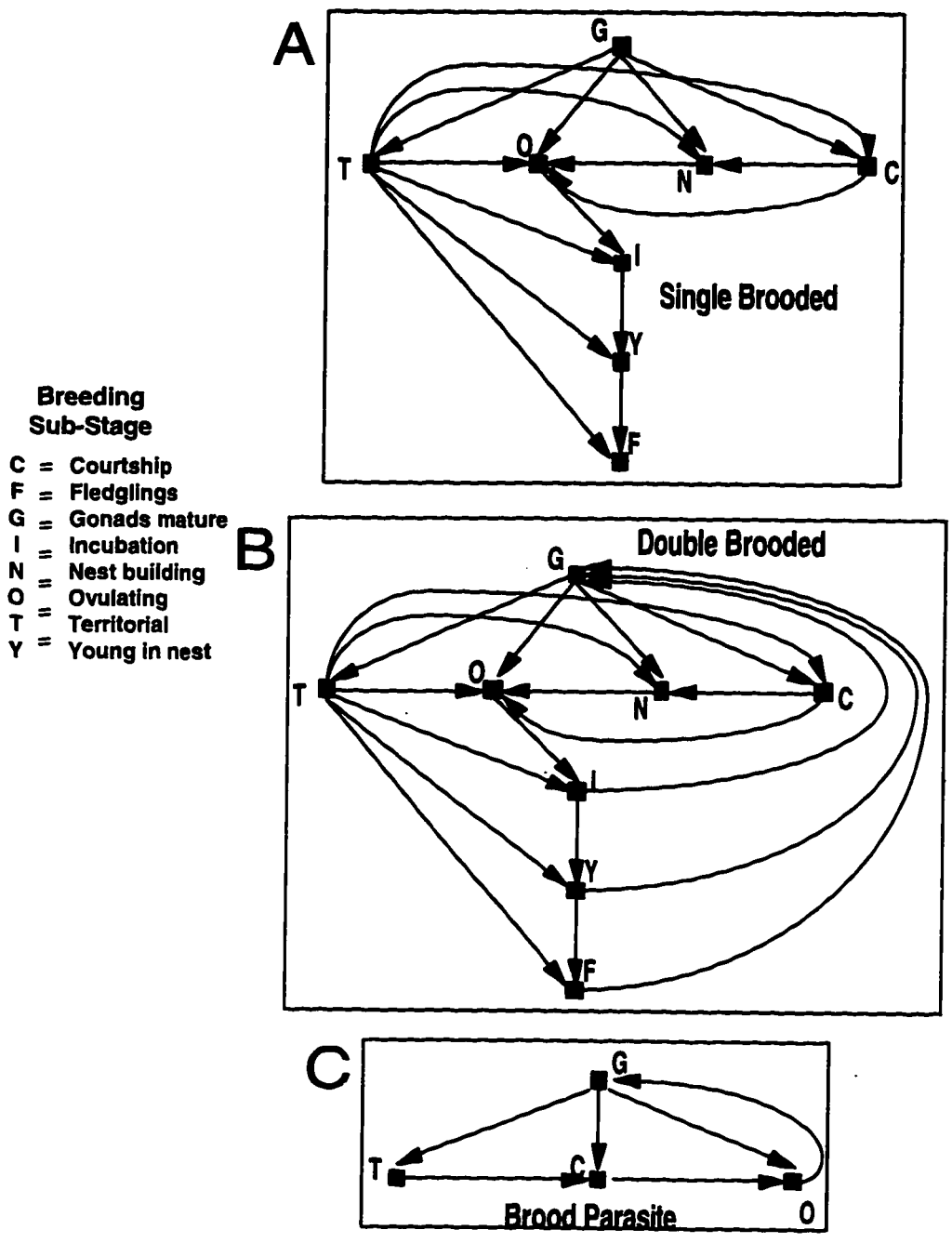


Figure 2.11 - Schematic representation of connectivity of sub-stages within a specific life history stage in different taxa of birds. In A, connectivity of single brooded populations when in the breeding stage is not as complex as in a double-brooded population (B). The breeding stage is even more simplified in terms of number and connectity of sub-stages in a brood parasite that has not parental sub-stages (C).

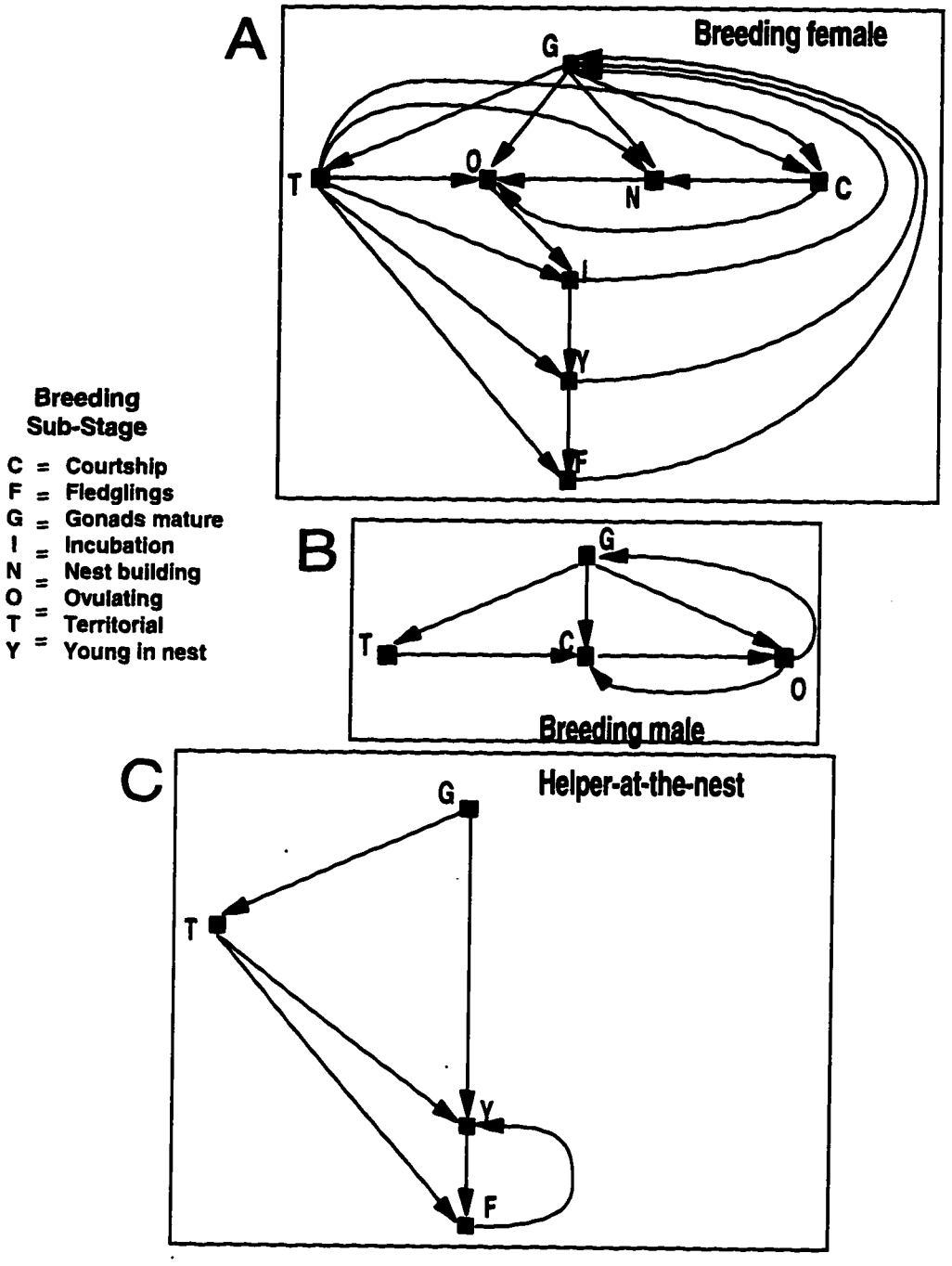


Figure 2.12 - Schematic representation of connectivity of sub-stages within a specific history stage within a population. In A, connectivity of sub-stages in a breeding female may be much more complex than for that of males which show no parental care (B). In cooperatively breeding species, the number of sub-stages and their connectivity may be even more simple. In this case individuals show some parental sub-stages, but no sexual sub-stages (C).

parental behavior and they show connectivity of 0.58 (Fig. 2.12b). In some cooperatively breeding species such as the white-crowned sparrow weaver, *Plocepasser mahali*, (Wingfield *et al.*, 1991) there is a dominant male and a dominant female who provide parental care and also have a sexual phase (connectivity thus equals 0.34 as in Fig. 2.12a), but there are also other individuals in the group that have no sexual phase but do feed young (Fig. 2.12c). Here there are only 4 sub-stages in the breeding stage with a connectivity of 0.5. These variations in numbers of sub-stages within the same phenotypic stage of individuals within a population and their connectivity could have fundamental influences on the neural and hormone control mechanisms. Such analysis as this may allow meaningful comparisons to be made when conducting experiments.

Implications for Hormone Control Mechanisms.

Endocrine secretions regulate many aspects of homeostasis as well as transitions in morphology, physiology and behavior in relation to predictable changes in the environment. Since most vertebrates live in changing environments, they change their physiological state to maximize survival at different times of year. These changes can be summarized as a finite state machine consisting of a temporal sequence of life history stages (phenotypic stages). The overall benefit of these transitions is to maximize life time fitness (see chapter one). However, the number of life history stages, their duration and the degree of overlap with other stages varies from species to species, and from

population to population. As hormones regulate the development of each state and its termination, then the n umber of stages and the degree of overlap with other stages could have a major influence on endocrine mechanisms and their control by environmental cues. For example, in Fig. 2.1 we see that the rock dove has only two life history stages with considerable overlap (Fig. 2.4) whereas the rock ptarmigan has seven with less overlap. The need for accurate timing of the transitions from one stage to the next in ptarmigan is clearly more important than in the rock dove. It is known that annual changes in photoperiod acting through neuroendocrine and endocrine secretions regulate at least some of the life history stages of the rock ptarmigan (*e.g.* Stokkan *et al.*, 1985, 1986). On the other hand, regulation of breeding and molt appear to be independent of photoperiod in the rock dove (Lofts and Murton, 1968). The analysis using diversity indices may help identify where possible differences, or similarities, in environmental and endocrine mechanisms may exist. The analysis provides an objective framework on which to make broad comparisons across taxa or even populations within taxa.

Each stage has a unique set of sub-stages that can be activated in complex sequences or combinations. Analysis of connectivity allows us to compare not only number of stages but also the complexity of interrelationships of that set of sub-stages. For example in figure 2.12 we can see that the neuroendocrine and endocrine mechanisms activating sexual sub-stages in a cooperative breeder, the white-browned sparrow weaver, may be in operation in dominant breeding males

and females but not necessarily in the helpers-at-the-nest that only express parental sub-stages. Does this mean that the helpers are incapable of expressing sexual sub-stages? Certainly it appears true that brood parasites such as the cowbird can activate sexual sub-stages (Fig. 2.11c) but have lost the ability to express parental sub-stages even if given hormones known to activate parental behavior in other species (see Dufty and Wingfield, 1986a,b).

Other comparisons across phenotypic stages show that these sub-stages are truly unique to their phenotypic stage. For example, in song sparrows, males in the winter stage (*e.g.* Fig. 2.10d) were unable to respond with sexual sub-stages typical of the breeding stage (Fig. 2.10a) when presented with females made sexual receptive experimentally. However, these males responded normally once increasing photoperiod had triggered the development of the breeding stage (Wingfield and Monk, 1994). Connectivity may be a useful way to compare complexity of expression of sub-stages in many other combinations of taxa or populations in diverse habitats. The potential for investigating endocrine mechanisms and environmental control is very high.

Chapter III:

Using a Finite State Machine Model to Understand Variations in Organismal Responses to Unpredictable Events in the Environment

Introduction

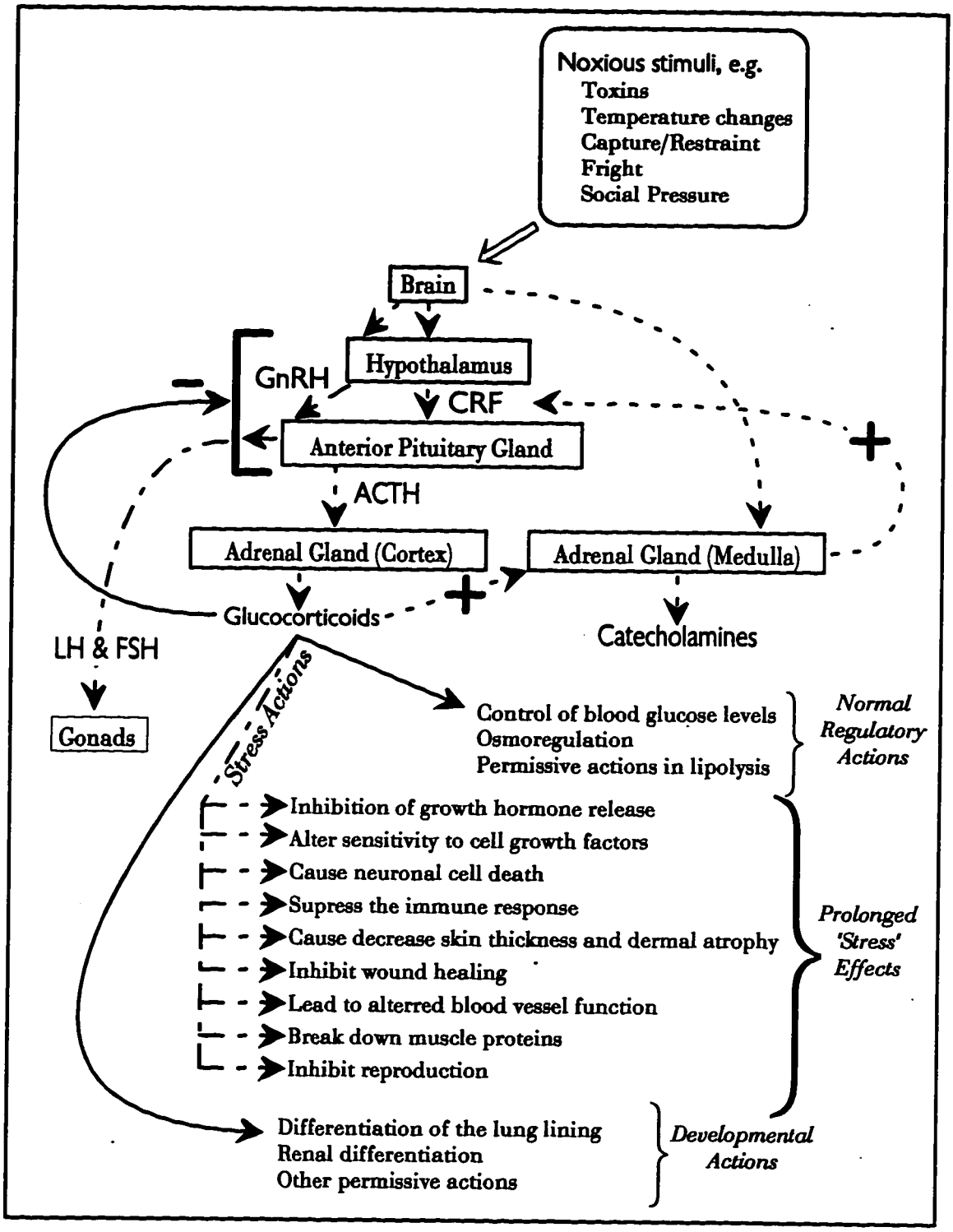
In the mid-1930's, Hans Selye studied the effects that a wide variety of noxious agents had on the physiology of mammals (see Selye, 1946 for a review). During this work on pathological and biochemical changes caused by noxious agents, he was

> *"struck by the fact that certain manifestations are always the same, irrespective of the specific nature of the eliciting damaging agent."* (Selye, 1946)

These "manifestations", which involve a cascade of hormones and a series of physiological changes within the animal (see figure 3.1 and Hadley, 1996), have been shown in every vertebrate examined. He called this suite of physiological responses to noxious stimuli the General Adaptation Syndrome (GAS) (Selye, 1936). Although Selye's extension of the GAS into the so called "diseases of adaptation" (Selye, 1946) proved to be wrong in many cases (see Munk *et al.*, 1984), his description of the generalized reaction of animals to "stressors" has formed the foundation for most work on stress in vertebrates.

This nearly stereotypical response to noxious stimuli is now called the "stress response". The stress response occurs in response to an extremely diverse

Figure 3.1 - Diagram of the response of vertebrates to noxious stimuli. The brain integrates responses to the stimuli, and triggers two hormonal responses: One leading to elevated catecholamines and the other to elevated glucocorticoids (GCs). Simulation of the adrenal medulla by the sympathetic nervous system leads to a nearly immediate increase in circulating catecholamines. These catecholamines, in addition to stimulating the "fight or flight" response, stimulate the hypothalamus to secrete Corticotropin Releasing Factor (CRF). At the same time, the hypothalamus is secreting CRF in response to the noxious stimulus. CRF passes through the hypothalamo-hypophyseal portal system to the anterior pituitary, where it stimulates the release of Adrenocorticotropic hormone (ACTH). ACTH the flows though the general blood supply and stimulates the adrenal cortex to release glucocorticoids (GCs). The GCs have many beneficial effects (such as the regulation of blood glucose and a role in osmoregulation), but prolonged exposure causes many negative effects. GCs can act on the hypothalamus to inhibit the releases of Gonadotropin Releasing Hormone (GnRH) and on the anterior pituitary to inhibit the release of Luteinizing Hormone (LH) and Follicle Stimulating Hormone (FSH) Several developmental processes also require the action of glucocorticoids.


Noxious stimuli, e.g.
Toxins
Temperature changes
Capture/Restraint
Fright
Social Pressure
Brain
Hypothalamus
GnRH
CRF
Anterior Pituitary Gland
ACTH
Adrenal Gland (Cortex)
Adrenal Gland (Medulla)
Glucocorticoids
Catecholamines
LH & FSH
Gonads
Stress Actions
Control of blood glucose levels
Osmoregulation
Permissive actions in lipolysis
Normal Regulatory Actions
Inhibition of growth hormone release
Alter sensitivity to cell growth factors
Cause neuronal cell death
Supress the immune response
Cause decrease skin thickness and dermal atrophy
Inhibit wound healing
Lead to alterred blood vessel function
Break down muscle proteins
Inhibit reproduction
Prolonged 'Stress' Effects
Differentiation of the lung lining
Renal differentiation
Other permissive actions
Developmental Actions

suite of noisome factors including infectious diseases (Judd, 1917), capture/restraint by a "predator" (e.g., Holmes and Phillips, 1976; Siegel, 1980; Harvey, 1984; Wingfield *et al.*, 1982), bacterial toxins (Olitzki *et al.*, 1942), sudden exposure to cold (Alexandrova and Farkas, 1992) or heat (Wang and Edens, 1993), traumatic shock (Best and Solandt, 1940) and X-rays (Moon *et al.*, 1940)). Virtually any facet of the environment for which the animal is either quantitatively or qualitatively not prepared can act as a stressor, and trigger a stress response. A number of studies have shown that the proper activation of the stress response greatly decreases mortality to some severe stresses (*e.g.*, Carey *et al.*, 1971; Finlay and McKee, 1982; Sibbald *et al.*, 1977). While the many aspects of the stress response are nearly stereotypical, there can be large differences in the responses of individuals to the same stress (e.g., Camp and Robins, 1988; Sapolsky, 1985; Sapolsky, 1986; Sapolsky, 1988; Swendsen, *et al*, 1995; Meaney *et al.*, 1993a; Meaney *et al.*, 1993b). Some of this variation is due to genetic differences in the individuals (*e.g.*, Flaherty and Rowan, 1989; Freund *et al.*, 1988; Satterlee and Johnson, 1988; Carsia and Weber, 1986; Gross and Siegel, 1985), but the rest of the variation appears to be associated with differences in the physiological condition and/or reproductive status of the animals (Wingfield *et al.*, 1994a, 1994b).

Munk and coworkers (1984) argued that the function of the stress response was not to increase the animal's resistance to stress, but rather as a feedback

mechanism to keep the animal's defense systems (immune system and inflammation) in check, such that they do not over-shoot and thus endanger homeostasis. While there is definitely a complex interaction between the immune system and the hypothalamo-pituitary-adrenal axis (see review by Bateman *et al.*, 1989), this safety-valve action is only a small portion of the reactions which occur in response to noxious stimuli.

In chapter one, I developed a finite state machine model for describing and understanding how animals respond to changes in their environment. The animal (as a finite state machine) conforms to pair of functions - the response function and the state function:

$$response_i = f_{response}(excitation_i, state_i)$$
$$state_{i+1} = f_{state}(excitation_i, state_i)$$

As these equations indicate, both the actions of the animal ($response_i$) and the state the animal will occupy during the next time interval ($state_{i+1}$) are based only upon the current state ($state_i$) of the animal, and the stimuli ($excitation_i$) during the previous time interval. In any finite state machine model, there are three kinds of variables: excitation, response, and intermediate. In the animal finites state machine model (from chapter I), the excitation variables are cues from the environment, and the response variables are the animal' s responses to those cues. The intermediate variables act within the machine (animal). While the intermediate variables were ignored in chapter one, they will be dealt with in

detail here as they are important in determining the state of the animal, and hence the animal's response to unpredictable events in the environment (see below). The effect of two intermediate variables (energy reserves and number of groups of progeny in a lifetime) on responses of individuals to unpredictable events in the environment will be examined.

Adult Responses to the Unpredictable

The sudden exposure of an animal to a noxious stimulus triggers a series of hormonal responses. The brain integrates cues from the environment, stimulating the adrenal medulla via the parasympathetic nervous system (Figure 3.1 and Hadley, 1996). The adrenal medulla begins releasing catecholamines into the blood. The brain also stimulates the hypothalamus to release Corticotropin Releasing Factor (CRF) from the median eminence into the hypothalamo–hypophysial portal system. CRF stimulates the anterior pituitary to release Adrenocorticotropic Hormone (ACTH) into the general bloodstream. Arginine vasotocin, Arginine vasopressin, and oxytocin can also be released from the median eminence to stimulate the release of ACTH (*e.g.*, Romero *et al.*, 1993; Meaney *et al.*, 1996). ACTH then stimulates the adrenal cortex to produce glucocorticoids. These glucocorticoids stimulate the mobilization of amino acids from muscle proteins for use in gluconeogenesis, and through permissive actions tend to stimulate release of free fatty acids from adipose tissue. Glucocorticoids also trigger behavioral changes in animals. The combined responses triggered by

the catecholamines and glucocorticoids help the animal to survive the "stressful" event (Dunlop, 1963). The glucocorticoids also shut down many processes not necessary to the emergency state. Inflammation and the immune system are inhibited, preventing the animal from being immobilized by its own defenses (Bateman *et al.*, 1989). Other energy demands, such as reproduction (Collu *et al.*, 1984) and growth (Slotkin *et al.*, 1992), are also inhibited by the glucocorticoids, providing more available energy to help maximize the animal's chances of surviving the emergency.

Though elevated glucocorticoids in the blood allow an animal to survive many "stressful" events, there is an extravagant cost for prolonged maintenance of high blood glucocorticoid levels. Prolonged exposure to elevated glucocorticoids results in a vast array of negative effects such as inhibition of reproduction (*e.g.*, Krulich *et al.*, 1974), inhibition of growth (*e.g.*, Slotkin *et al.*, 1992), suppression of the immune response (*e.g.*, Dohms and Metz, 1991; Irwin, 1994; Wilckens, 1995), neuronal cell death (*e.g.*, Virgin *et al.*, 1991), and wasting of muscles (*e.g.*, Dardevet *et al.*, 1995). With such extreme costs for extended exposure to elevated glucocorticoids, the "stress" response system only increases the fitness of the animal during relatively short-term disturbances, and is detrimental to the animal during protracted challenges to homeostasis.

Terminology

Unfortunately, the term stress has been applied to an extremely broad

range of physiological states such as exercise stress, stress of reproduction, fasting stress, emotional stress, and stress from noxious stimuli. Though there are some similarities in the physiological responses to all of these stresses, they are clearly not identical. There appear to be different pathways for stimulating the stress response, depending on the nature of the stress (see Romero *et al.*, 1993; Meaney *et al.*, 1996). Exercise stress, unless it is excessive and beyond the animals ability to respond, does not endanger the animal's homeostasis. Birthing stress, while extreme in humans, is a normal physiological process, and is prepared for by the animal during pregnancy. In some animals, such as Emperor Penguins, fasting is also a normal part of the animal's breeding season and is prepared for with the deposition of fat (Stonehouse, 1953). In most other animals however, fasting is not self-imposed and not necessarily predictable, and thus has very different physiological responses. Emotional stress, and the stresses from noxious stimuli such as cold, excessive heat and infections are either not predictable, or are not within the animal's ability to control its environment.

As the word stressor and the concept of stress have become associated with a confused morass of different meanings, we need to have a new set of terms which define specifically the emergency response to challenges to homeostasis. These challenges are normally short-lived under natural conditions (or the animal will die, or move away from the challenge), and they stimulate an emergency state within the animal which is not within the normal progression of physiological

states (see chapter I for an in-depth discussion of physiological states). I will use the term *Transient Perturbation Factors* (TPF) instead of stressors, since TPF implies that the factor is of limited duration, and that it perturbs the normal cycle of physiological states in the animal's life cycle. A *Transient Perturbation Factor* will be defined as any noxious stimulus to which the animal is either qualitatively or quantitatively not adapted in its current physiological state, and the occurrence of the stimulus is either unpredictable, or the animal cannot normally express an appropriate response to the stimulus. It is important to keep in mind that even though a stimulus seems noxious to an investigator, it may not be perceived as such by the study animal. The noxious stimulus must be outside the normal range of the animal's ability to cope with the stimulus. A human with a body temperature of 43°C would soon die, but this same "excessive" temperature is often observed in desert iguanas (*Dipsosaurus dorsalis*) when active during the heat of the day (Swezey and Somero, 1982) and produces no ill effects in these lizards.

An animal's ability to cope with a particular stimulus can vary over time. For example, a temperature of 4°C is not a noxious stimulus to goldfish acclimated to cold temperatures during the winter, but it is often lethal to goldfishes acclimated to warm summer temperatures (Fry *et al.*, 1942). The stimulus must either be unpredictable, such as a sudden storm, or capture by a predator, or the stress is uncontrollable by the animal (that is it has no way of dealing with the stimulus). If the noxious stimulus is predictable, and the animal can "prepare"

itself, either by alterations in behavior or physiology, the stimulus ceases to be a TPF. Some stimuli are uncontrollable even if they are predictable. Social stress in a baboon troop may be relatively predictable by the lower order males, but as they are usually smaller and/or weaker than the dominant male, they can not change their social status (Sapolsky, 1982; Sapolsky, 1983; Sapolsky, 1990). Their only possible active response is to leave the troop, and this is not a viable option. Many experimental stress experiments fall into this uncontrollable class: the stimulus may occur at predictable intervals, but there is no way for the animal to deal with the stimulus (*e.g.*, electric shock, hypertonic saline injections).

In keeping with the transitory nature of the stimulus and the response, the animals will be said to adopt a *Transitory Emergency State (TES)* in response to transient perturbation factors. A TES is characterized by the inhibition of systems not "required" during a short duration emergency (such as the immune system and the reproductive system), the mobilization of some energy reserves, and the alteration of behavior to those appropriate to dealing with the TPF. These alterations in the physiological state of the animal often allow the animal to cope with the emergency situation. A TES is obtainable from almost any physiological state in the animal's life cycle and by its very nature interrupts the normal progression of physiological states.

Every system within an organism has some sort of effect on the survivorship of the organism. The *adaptive value* for a system will be defined here as the effect

the system has on the survival of the animal, and on its potential for production of offspring. This adaptive value may be zero, or possibly slightly negative, as in the case of the human appendix which has no apparent function (see Romer, 1962). Most systems within the animal will have high, positive adaptive values. Obviously organ systems such as the circulatory system have adaptive values which are large and positive, their benefit greatly outweighing the cost of producing and maintaining the organ system.

The adaptive value for a system can change over time. In the larvae of an aquatic salamander, the gills have a large positive adaptive value as they greatly increase the larva's ability to extract oxygen from the water. If these gills were to persist in the terrestrial adult, the fact that the gills could be easily damaged, could increase the animals water loss rate and are essentially useless as a terrestrial respiratory organ would give the gills a negative adaptive value in the adult stage. The adaptive value of bright plumage and showy displays may be high for a bird during the breeding season, but flamboyance has a tendency to attract predators and will generally have a negative adaptive value during the non-breeding seasons.

If the positive adaptive value of a system drops enough it will reach a *critical adaptive value*, where it has neither a positive or a negative effect on the fitness of the animal. If the adaptive value for the system drops below the critical adaptive value, the system is reducing the fitness of the animal. If the adaptive value for a system is above the critical adaptive value the system is increasing the

fitness of the animal. The critical adaptive value will act as an intermediate variable in the finite state machine model for animals, as developed in chapter one, and will be pivotal in determining how animals respond to TPFs.

In natural conditions in the wild, an animal will exist in one of a finite number of physiological states (see chapter one) which are determined by the animal's starting state, and the signals/cues it has received. Responses to unpredictable events will follow these same state-dependent rules. When the animal is exposed to a TPF, it will attempt to adopt a TES based on the state of the animal before the TPF, and appropriate to the set of cues at that time (see figure 3.2). The cues that determine the particular TES adopted include the normal environmental cues (such as initial predictive cues and supplementary cues; see chapter one), as well as the TPF. The TPF can over-ride the effects of many of the other cues, but the normal cues can be important in determining the particular TPF adopted. For example, if food seed supplies in the territory of a bird are large (a supplementary cue), the bird may choose to stay on its territory during a snow storm if it can collect enough seeds through the snow cover, but may abandon its territory if the food supplies were poor. If the animal can not attain a proper TES or the TPF is too severe, the animal will die. The normal environmental cues are certainly important in determine where the animal will re-enter the normal cycle of physiological states when the TPF is removed. When the TPF is removed, the animal will re-enter the normal progression of states at a state

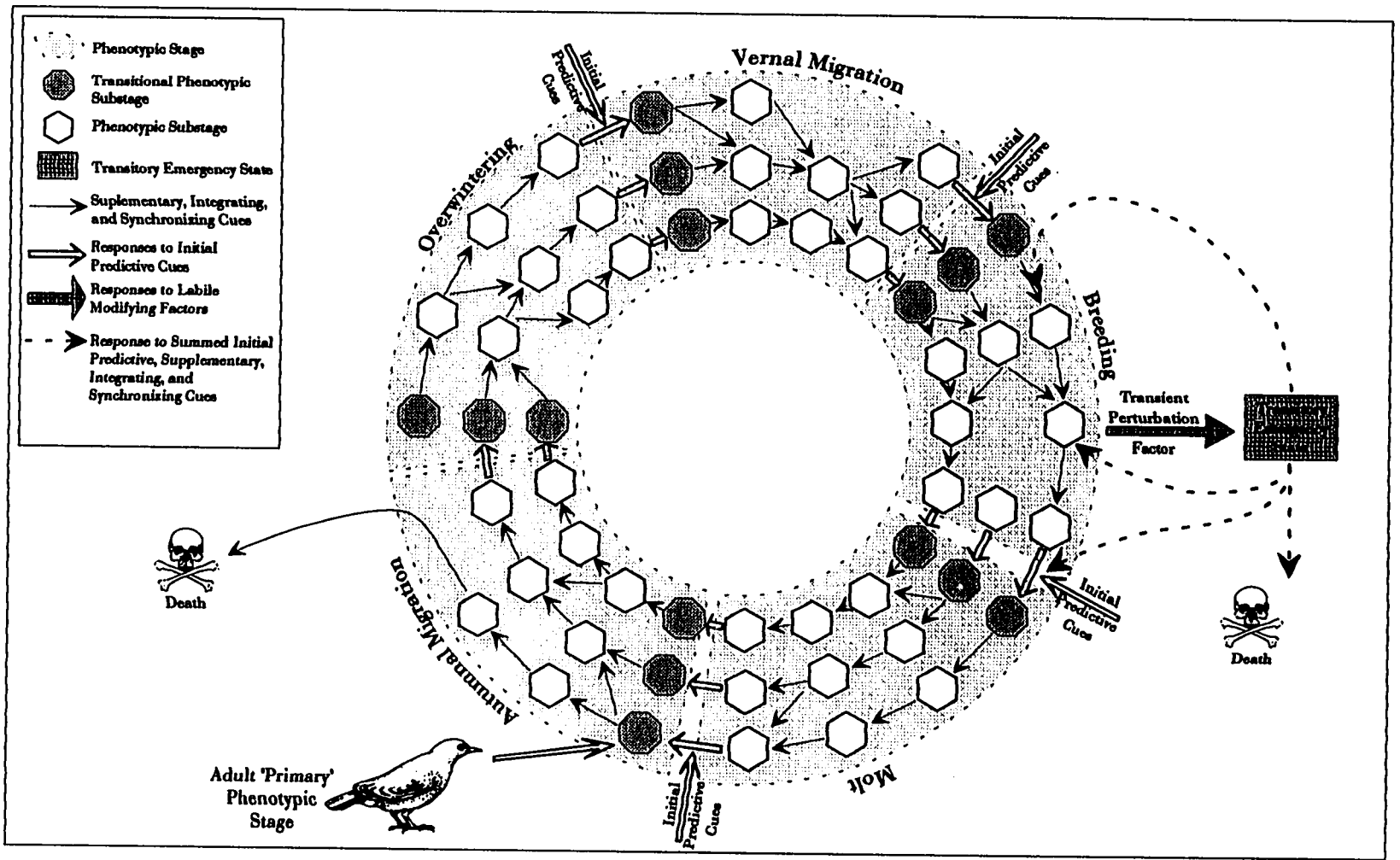


Figure 3.2 - A finite state machine model of the life cycle of an adult bird showing the effects of transient perturbation factors (TPFs). Death can occur at any time during the cycle. A TPF will normally shunt an animal from the normal cycle of states into a transitory emergency state. After the TPF is removed, environmental cues determine the state the animal will express next. If the TPF is too intense, or is not removed, the animal will die.

determined by the environmental cues the animal receives at that time and the TES the animal is in at the time the TPF is removed (including the effect the TPF had on the animal's state). If for instance, a bird was feeding young at the time of a severe storm, and the nest was destroyed, the bird can not return to the state of feeding the young when the storm passes. It is early enough in the season (that is the cues are still correct for breeding) the bird can renest and produce another brood, but if it is too late in the year for breeding, the bird will begin to molt its nuptial plumage, preparing for the autumnal migration. If the storm did not harm the nestlings or the nest, the bird could return to its original state (assuming it was not other wise injured), and continue to feed the young.

The situation may be very different in laboratory conditions (see figure 3.3). Under natural/field conditions (see Figure 3.3, upper panel) an animal can be in a fairly large number of states triggered by a variety of environmental cues. Exposure to TPF will trigger a one of many possible TESs (or the animal may move away from the TPF). In the laboratory situation (Figure 3.3, lower pane), most of the cues an animal would receive in the wild have been eliminated, and hence the number of possible physiological states available to the animal is reduced (as the state adopted during the next time interval is determined by the current state, and the stimuli detected during the current time interval). Some of these possible laboratory states may not normally exist in the wild as they result from cues not normally available in the wild. Animals in the lab are usually in

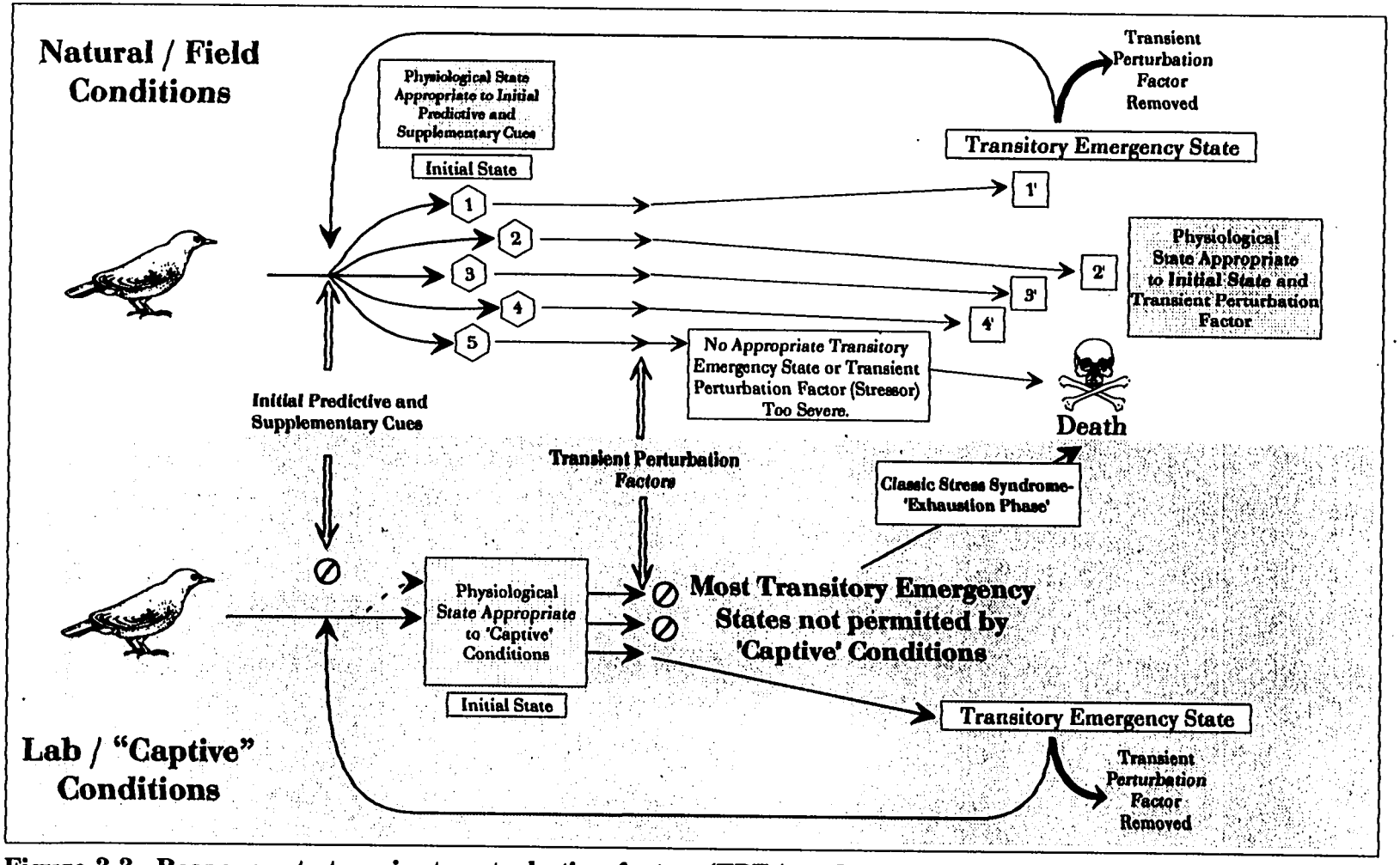


Figure 3.3 - Responses to transient perturbation factors (TPFs) under natural and laboratory conditions. At any time, an animal will be in some physiological state determined by its previous state, and the cues it has received. When exposed to a TPF, it will normally alter its state into one of the transitory phenotypic states in order to respond to the TPF. Lab conditions limit most responses, and the animal usually cannot respond appropriately.

cages, so this alone is a major change in their environment, and hence a major change in their physiological state. Thus when exposed to a TPF, the possible reactions of the laboratory animal are severely constrained. The animal can not move away from the stress, and the constraints imposed on the animal by laboratory conditions severely limits the possible TESs available to the animal. Such limitation of the possible TESs can lead to death, as it may not be possible to adopt an appropriate TES for the particular TPF. The laboratory TPF itself may not exist in the wild, and hence the animal may have no way of dealing with the TPF. The life span of many animals kept in captivity is greatly reduced, possibly resulting from "unnatural" TPFs which lead to prolonged exposure to elevated glucocorticoids, or from TPFs which could be dealt with in the wild, but cannot be coped with in captivity due to limitations imposed by captivity on the possible TESs.

Modulation of the TPF response during the breeding season.

Breeding is the most crucial phase of an organisms life cycle, at least in evolutionary terms. Since fitness can be considered a function of the number of offspring produced into the next generation, the breeding season is pivotal to the animal's fitness. Maximizing lifetime reproductive success is very important to all organisms, so physiological process which interfere with breeding are usually inhibited.

The TPF response can inhibit reproduction, either by altering behavior , or inhibiting the hypothalamo-adenohypophysial-gonadal axis (Collu *et al.*, 1984; Greenburg and Wingfield, 1987). The number of possible attempts an animal has to breed in a lifetime is an important intermediate variable, and one that can regulate responses to TPFs. In short lived vertebrates, which may produce only a single group (brood, litter, etc.) of offspring, there will be a large reduction in the adaptive value of the TPF response during the breeding season as the TPF response would inhibit the one chance they have at producing offspring (see Figure 3.4, bottom panel). This drop in adaptive value may be large enough to cause the adaptive value to drop below the critical adaptive value. In such situations, the TPF response would either be inhibited or muted, so as not to interfere with the animal's limited chance for reproduction. Vertebrates which produce many sets of offspring in a lifetime will have only a small drop in the adaptive value during the breeding season (see figure 3.4, top panel). By plotting the effect on the adaptive value of a TES versus both time of year and the number of broods in a lifetime, we get a landscape as show in Figure 3.5. For most of the year, there is no effect of the number of broods in a lifetime on the adaptive value of the TES. Yet during the breeding season we see a distinct valley, becoming increasingly deeper as the number of lifetime broods decreases. At low numbers of lifetime broods, the adaptive value of the TES drops below the critical adaptive value, indicating that animals with low number of lifetime broods may suppress their response to TPFs

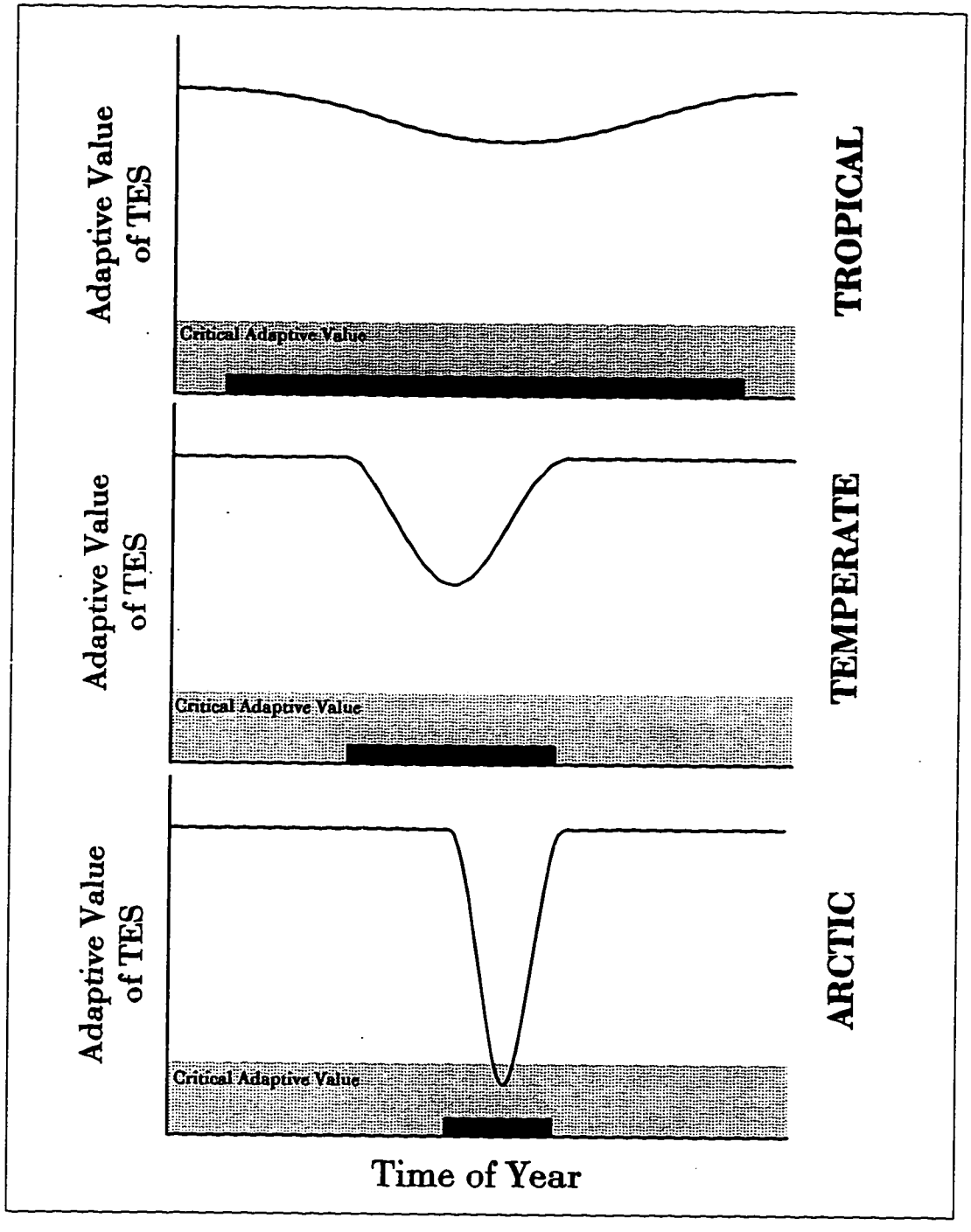


Figure 3.4- Model of breeding season depression in the adaptive value of a transitory emergency state. Going from low latitudes to high latitudes, there is a decreasing length of the breeding season (black bars). As the length of the breeding season decreases, the adaptive value of expressing a TES during the breeding season decreases.

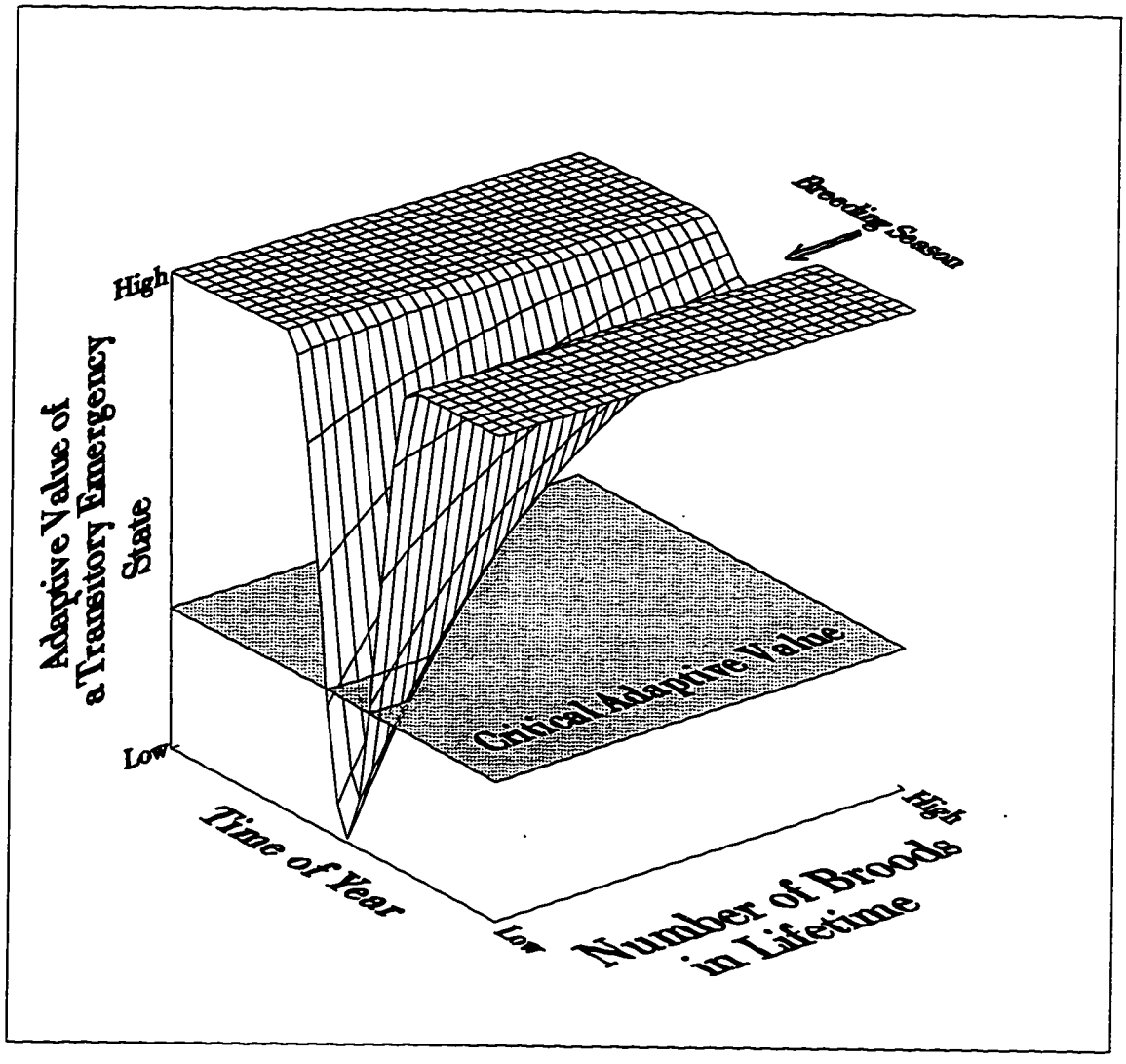


Figure 3.5 - Effect of time of year and number of broods in a lifetime on the adaptive value of a transitory emergency state (TES). The grey plane represents the critical adaptive value. During most of the year, the number of lifetime broods has no effect on the adaptive value of the TES. During the breeding season, there is a very large pressure to complete reproduction if the number of lifetime broods is small.

during the breeding season.

Of course, simply the number of offspring in a life time will not be the sole determining factor determining an animal's response to a TPF. The response of an animal to a given stimulus (including TPFs) will be determined by the animal's current state, and the integration of all stimuli at that time. One intermediate variable important in determining an animal's physiological state is the energy reserves of the animal. High energy reserves help an animal in a number of ways, including providing energy for such tasks as reproduction, growth, and migration. Energy reserves can also help an animal survive many TPFs (unpredictable emergencies).

We can visually examine the interaction two variables have on the adaptive value to a TES by creating three dimensional graphs. Plotting a series of these 3-D graphs along a series of a third variable gives us a feel for how three different variables interact. We will look at the effect an intermediate variable (energy reserves) and an excitation variable (time of year) have on the adaptive value of a TES in short-lived birds, and compare these effects in environments of different latitudes. As an animal's energy reserves increase, the need to adopt a TES in response to many TPFs becomes less, as the animal is better able to "sit-out" the emergency. For an animal with a moderate number of lifetime broods, this slight drop in the adaptive value associated with elevated energy reserves does not bring the adaptive value any where near the critical adaptive value, even when coupled

with the decrease in adaptive value associated with the breeding season, and hence has little or no effect on the expression of a TES (Figure 3.6). If however, the animal has only a relatively small number of lifetime broods, this slight drop in the adaptive value of a TES due to elevated energy reserves also has minimal effects on the adaptive value of the TES for most of the year, but may cause the adaptive value of the TES to drop below the critical adaptive value during the breeding period (see Figure 3.7). In such a situation, animals with low energy reserves would show the normal response to a TPF during the breeding season, and those with high energy reserves would show a muted response (or possibly no response) to TPFs during the breeding season.

Example I. Attenuation of the TPF response in high latitude breeding birds.

Birds which migrate to Arctic regions in order to breed are under fairly severe time constraints. They must breed early enough in the summer such that the resulting offspring are fledged and ready for the autumnal migration. This time constraint means that during years when spring comes late, the birds must begin setting up territories while there is still a strong possibility of bad weather and snow fall. There is only a very narrow window where re-nesting is possible if a brood is lost, as it soon becomes too late in the year to produce offspring which will survive and be able to migrate. Both of these constraints may act to reduce the adaptive value of the TPF response during the breeding season.

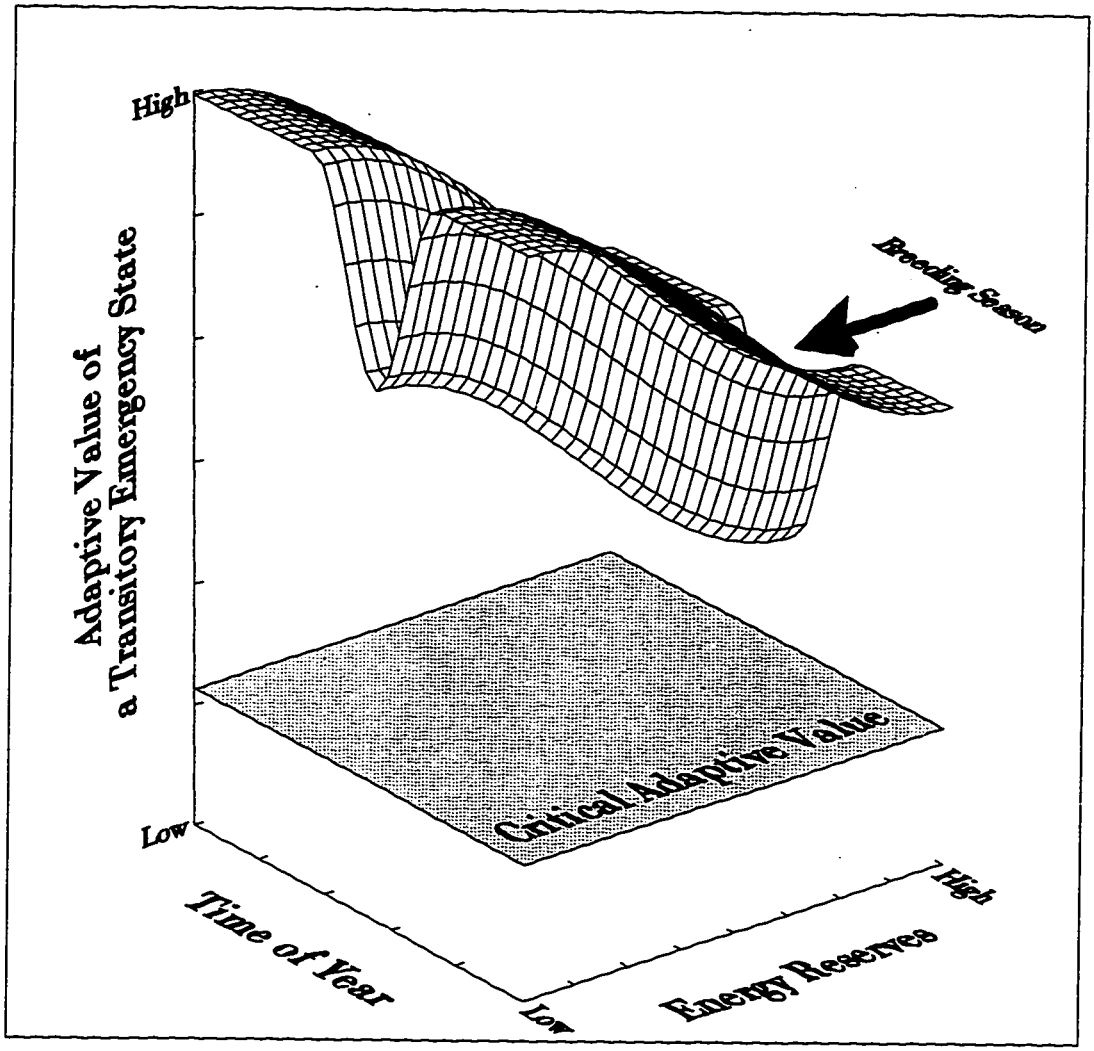


Figure 3.6- Model of the effect of time of year, and energy reserves on the adaptive value of expressing a transitory emergency state (TES) in a short lived animal that breeds in the temperate zone. The grey plane represents the critical adaptive value. The breeding season is shorter than in the tropics, but still long enough normally for multiple broods. There is a larger depression in the adaptive value of the TES associated with the breeding season than in the tropics, but it is still normally above the critical adaptive value.

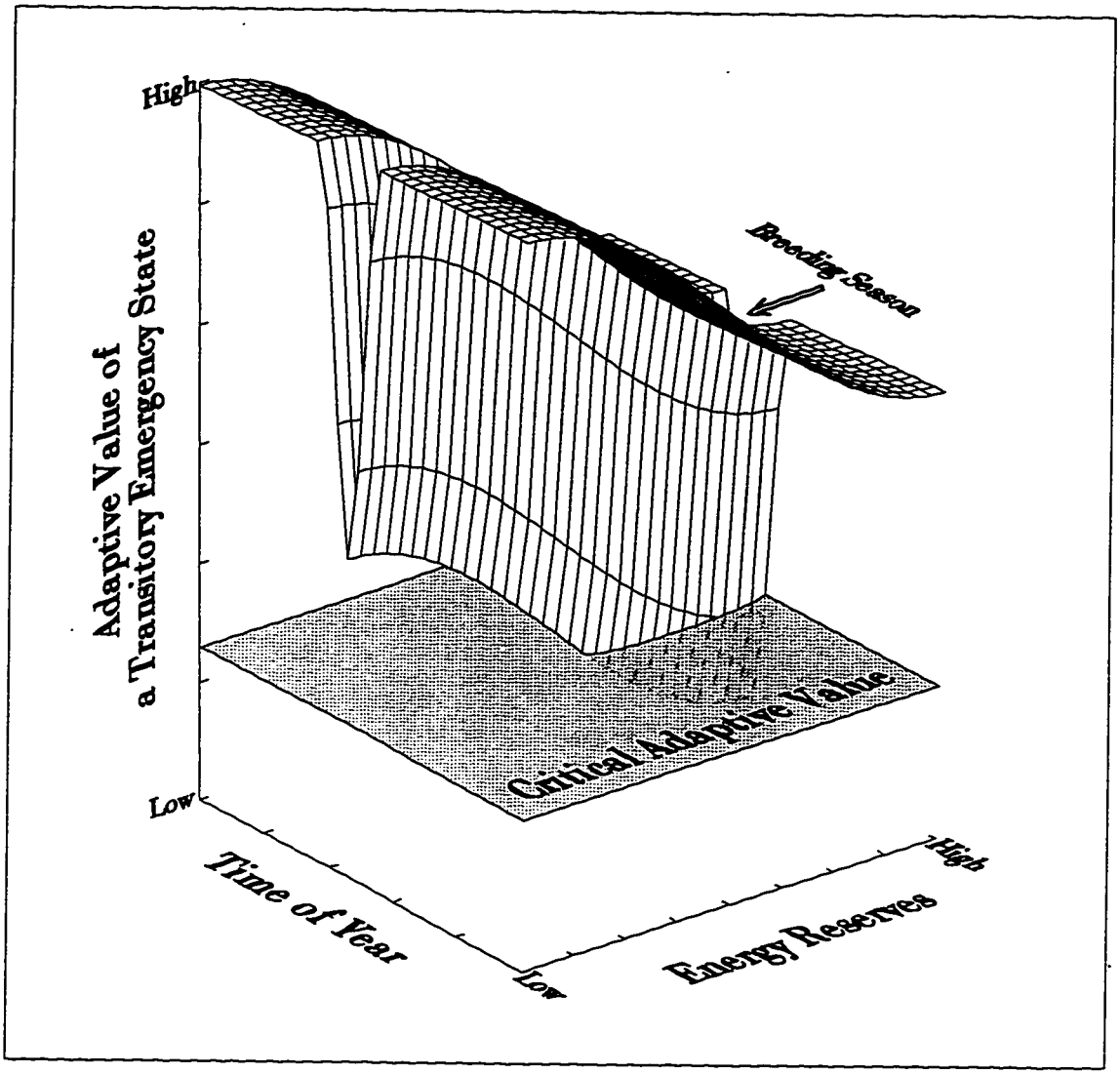


Figure 3.7- Model of the effect of time of year, and energy reserves on the adaptive value of expressing a transitory emergency state (TES) in a short lived animal that breeds at high latitudes. . The grey plane represents the critical adaptive value. The breeding season is shorter than in the tropics, but still long enough normally for multiple broods. There is a larger depression in the adaptive value of the TES associated with the breeding season than in the tropics, but it is still normally above the critical adaptive value.

Adopting a TES during usually leads to the elevation of plasma glucocorticoid levels, a conditions which in normally inhibitory to both reproductive behavior, and maintenance of the reproductive system (Greenburg and Wingfield, 1987). If the gonads begin to regress in response to elevated glucocorticoids, it may take on the order of weeks to recrudesce the gonads, making it too late in the season for beginning to re-nest. As the environmental cues may no longer be correct for gonadal recrudescence, it may not be possible to reverse the regression process once it has begun. Thus is should be in the interest of these small, short-lived, migratory birds to either repress the stress response, or limit the magnitude of the response such that it does not endanger their reproductive condition.

Common Redpolls (*Carduelis flammea*) are small finches which live in Arctic and sub-arctic regions during the entire year. When these birds breed north of the arctic circle, they are under the same time constraints as the long distance migrants during the breeding season. Throughout the entire year, Redpolls show a significant response to TPFs (as indicated by elevated plasma corticosterone levels; Wingfield *et al.*, 1994a). However, redpolls breeding near the north coast of Alaska at Barrow (71°N) showed a muted response to capture stress, when compared with redpolls breeding in the more southerly area around Toolik Lake (69°N). As expected (see figure 3.7), individuals at Barrow with larger fat reserves had smaller adrenocortical responses to capture and restraint (a TPF). It is

surprising that the redpolls breeding a Toolik Lake did not appear to down regulate their response to TPFs during the breeding season. This may be explained in part by the fact that day-time temperatures at Barrow Alaska are 10-20 C° lower than at Toolik Lake, causing the snow to melt off later in the summer, making the window for breeding and raising young even shorter than at Toolik Lake. Also surprising was that redpolls sampled during the winter at Fairbanks, Alaska (64°N) showed very different adrenocortical responses between January and February. In January, when the temperatures were very cold (-35°C), the response was very muted, similar to or less than the response at Barrow during the breeding season. However, the response of Redpolls during January when temperatures were a bit warmer (-15°C) showed adrenocortical responses to capture which were similar to, or larger than the responses of redpolls breeding at Toolik Lake. Such a rapid change (samples in January were taken only 16-18 days before the samples taken February) indicate that redpolls in the winter are operating in conditions where the adaptive value for the TPF response is near the critical adaptive value, as it appears that minor changes in the environment (such as changes in temperature) can alter the bird's responses to capture. Possibly winter conditions at Fairbanks are unpredictable enough that the unpredictability of the environment is lowering the adaptive value of the TPF response to near the critical adaptive value (see figures 3.8 and 3.12 and discussion of predictability below).

Snow buntings (*Plectrophenax nivalis*) and Lapland longspurs *(Calcarius*

lapponicus) are two passerine species which are abundant throughout the Arctic region during the breeding season. Since both these species arrive on their Arctic breeding grounds during the early spring when the weather may be quite severe, and since the Arctic summer is so brief, both of these species would be expected to show reduced responses to TPFs during the breeding season. When Lapland longspurs and snow buntings were examined at Barrow Alaska (71°N), neither of these birds showed the expected inhibition of the response to capture and restraint when compared with post breeding adrenocortical responses in the fall (Wingfield *et al.*, 1994b). Both longspurs and snow buntings showed individual variation to the TPF (capture and restraint), and this variation is inversely correlated with fat reserves: those with high fat reserves showed lower maximum responses to the TPF. While they show significant responses to capture and restraint, the magnitude of the increase in plasma corticosterone was 2-8 times less than that shown by white-crowned sparrows (Wingfield *et al.*, 1992). Lapland longspurs can show much stronger responses to TPFs during the breeding season (up to a 10 fold increase in corticosterone levels) when they have been exposed to a 3 day snow storm (Astheimer *et al.*, 1992). The inverse relationship of fat reserves with the maximal response to a TPF, and the ability to rapidly change (within 3 days) their response to TPFs indicate that both snow buntings and Lapland longspurs are operating with their adaptive value for the TPF response near the critical adaptive value during the breeding season.

Example II. Attenuation of the TPF response of some desert birds during the breeding season.

Birds breeding in the lower Sonoran zone of Arizona during the summer months can experience periods of intense heat, reaching up to 50°C, with limited or no access to open water (Wingfield *et al.*, 1992). High temperatures have been shown to elevate corticosterone levels in domesticated birds such as chickens and turkeys (Siegel, 1980; Harvey *et al.*, 1984). High temperatures, coupled with drought have been shown to elevate plasma corticosterone levels in bobwhite quail (Cain and Lien, 1985). While desert birds are undoubtedly more resistant to heat than turkeys, chickens, or bobwhite quail, they may still be exposed to unusually high temperatures during the summer which could disrupt breeding by triggering a TES. The length of the breeding season in the Sonoran desert does not put the same constraint on breeding as it does in the Arctic. However, the need to continue with reproduction during inclement heat waves may cause these birds to mute their responses to TPFs.

Wingfield *et al.* (1992), examined three species of Sonoran desert birds (black-throated sparrow, *Amphispiza bilineata*, cactus wren, *Campylorhynchus brunneicapillus*, and curve-billed thrasher, *Toxostoma curviostre*) during both the breeding season (summer) and in the winter. Adrenocortical responses to capture and restraint for these birds was lower in the breeding season than in the winter. There was no correlation of basal corticosterone levels with air temperature, so the

TPF response system in the summer birds was not already "maxed-out". Thus the birds breeding in the lower Sonoran zone have an adaptive value for the TPF response which is near, or below the critical adaptive value during the breeding season (similar to the lower panel of figure 3.4, only with a broader breeding season). Two other birds (Inca Dove, *Scardafella inca* and Albert's towhee, *Pipilo alberti*) that breed in riparian habitats that are much more benign than habitats in the lower Sonoran zone were also examined in this study. These riparian birds showed the same adrenocortical responses to capture and restraint in both the breeding season and during the winter. The relationship of adaptive value of a TES to time of year would look like the middle panel in Figure 3.4.

Example III. The response of King Penguins (Apentenodytes patagonica) to the "stress" of fasting.

King penguins breed on peripheral Antarctic islands, and on islands in cool temperate Subantarctic regions (Stonehouse, 1970). Both adult and juvenile King Penguins must survive prolonged periods of fasting. Each year, after a period of one or two week period of fattening at sea, the adults come ashore and begin a four to five week pre-nuptial molt. During the molt, these penguins do not feed, and normally loose up about 50% of there pre-molt body mass (Stonehouse, 1956; Weimerskirch *et al.*, 1992). After completing the molt, they return to the sea for two to four weeks for a second period of fattening. After fattening, the birds return to land and begin a three to six week courtship, where neither the male nor female

feeds. After the single egg is laid, the females return to the sea to feed, and the male continues to fast while incubating the egg for the first watch. After 12-15 days, the female returns and takes over incubation of the egg and the male begins a period of feeding at sea. The pair trade off periods of incubation/fasting and feeding till the chick hatches. After hatching, the chick is fed regularly for 3-4 months, reaching 10-12 kg (Cherel *et al.*, 1987). During the austral winter, food supplies drop and feeding of the young becomes sporadic. The chicks must survive prolonged periods of fasting during the winter, loosing up to 50% of their body mass. The chicks can tolerate a 4-6 month fast and up to a 70% loss in their body mass (Cherel and Le Maho, 1985).

The prolonged fasting periods of the King Penguins may seem very stressful (and extremely so in the case of the chick), yet there is normally no increase in plasma glucocorticoids during most of the fasting period (Cherel *et al.*, 1987; Le Ninan *et al.*, 1988). These birds (and the chicks in particular) are well adapted to predictable periods of fasting. Thus these prolonged fasts normally are not a TPF since they are both predictable and within the normal capacity of the animal to survive. The fasting can become a TPF if it is prolonged. If either parent is killed during the incubation period, the other parent will attempt to continue the incubation until its exceeds its normal capacity for fasting, at which point plasma glucocorticoids become elevated. These elevated glucocorticoids are indicative of a TES and may trigger a shift in the behavior of the adult penguin, shifting behavior

from brooding of the egg to feeding of the adult, causing the egg to be abandoned.

Modulation of the TPF response at the extremes of predictability

The TPF response system has been found in all organisms examined so far, from a wide variety of environments, so in most situations the TPF response system probably has a high adaptive value. There may however be environments where the TPF response drops below the critical adaptive value.

In a completely predictable environment, there would be no *transient perturbation factors* ("stressors"), as there would be no unpredictable events and thus no need for a response system to TPFs (*i.e.*, low adaptive value for the TPF response; see right side of figure 3.8). As the predictability of the environment decreases, the adaptive value of the TPF response increases, and probably reaches a plateau (see center region of Figure 3.8). As the predictability continues to decreases, the frequency of TPFs increases. Each TPF leads to an increase in glucocorticoids released in response to the TPF (bell shaped curves in Figure 3.5). If the frequency is relatively low, there will only be a small increase in the average blood glucocorticoid levels(Figure 3.9 - dashed line, upper panel). As the frequency increases, the average glucocorticoid level will be significantly above basal levels (Figure 3.5 - dashed line, middle panel). If the frequency becomes high enough, the average glucocorticoid will be at the maximum level (Figure 3.9 - dashed line, lower panel). In the latter two cases, the animal will begin to suffer

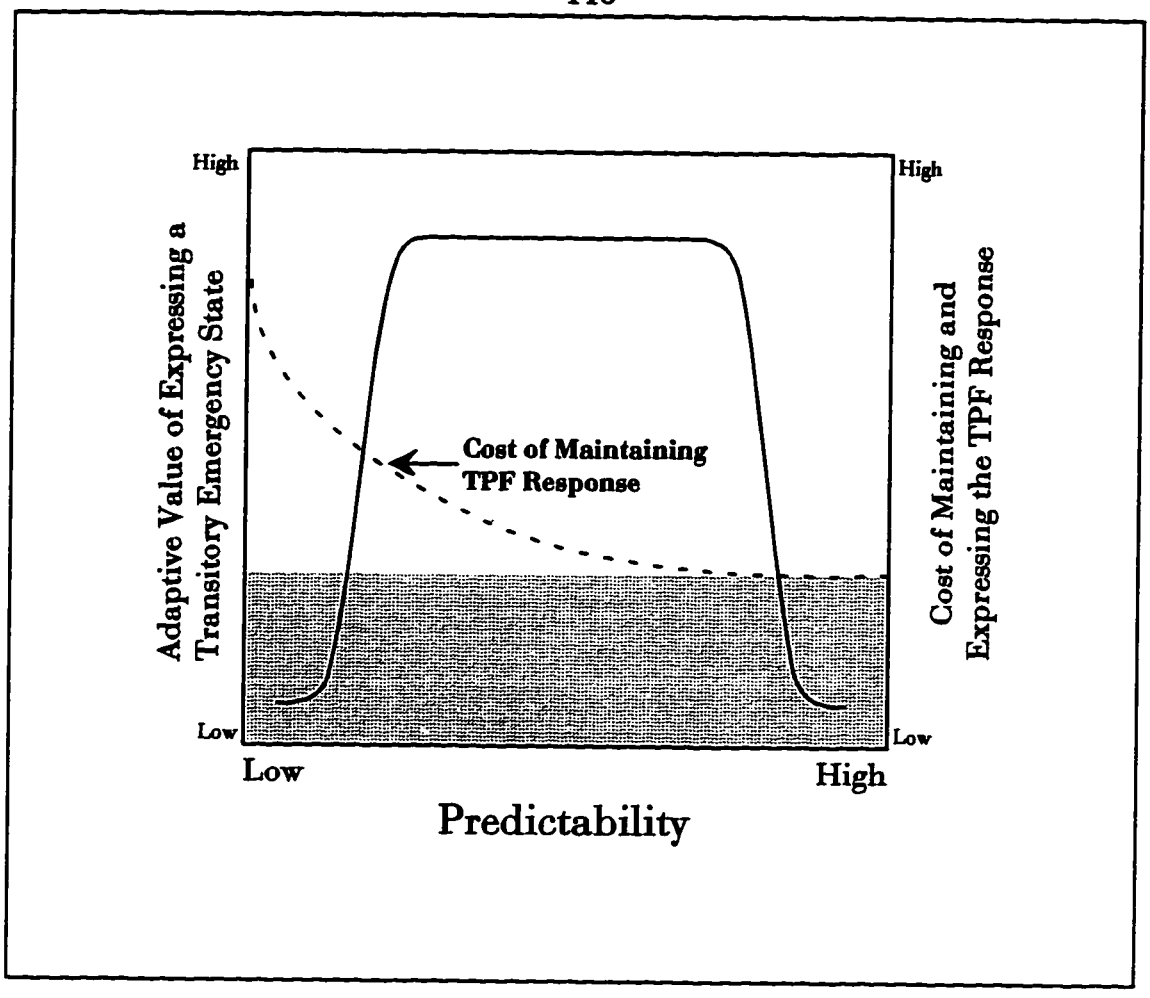


Figure 3.8 - Model of the effect of environmental predictability on the adaptive value of expressing a transitory emergency state (TES). Grey region represents the cost of maintaining the TPF response machinery (basal cost), and the dotted line represents the basal cost plus the cost of expressing a transitory emergency state. The thin solid line is the adaptive value of expressing a TES. At high predictability levels the TPF response is rarely, if ever expressed, and thus may have low adaptive value. At low predictability levels, the rate of TPFs is so high that the cost of expressing TES becomes too high. At both low and high predictabilities, the cost of maintaining the TPF response exceeds the adaptive value of expressing a TES.

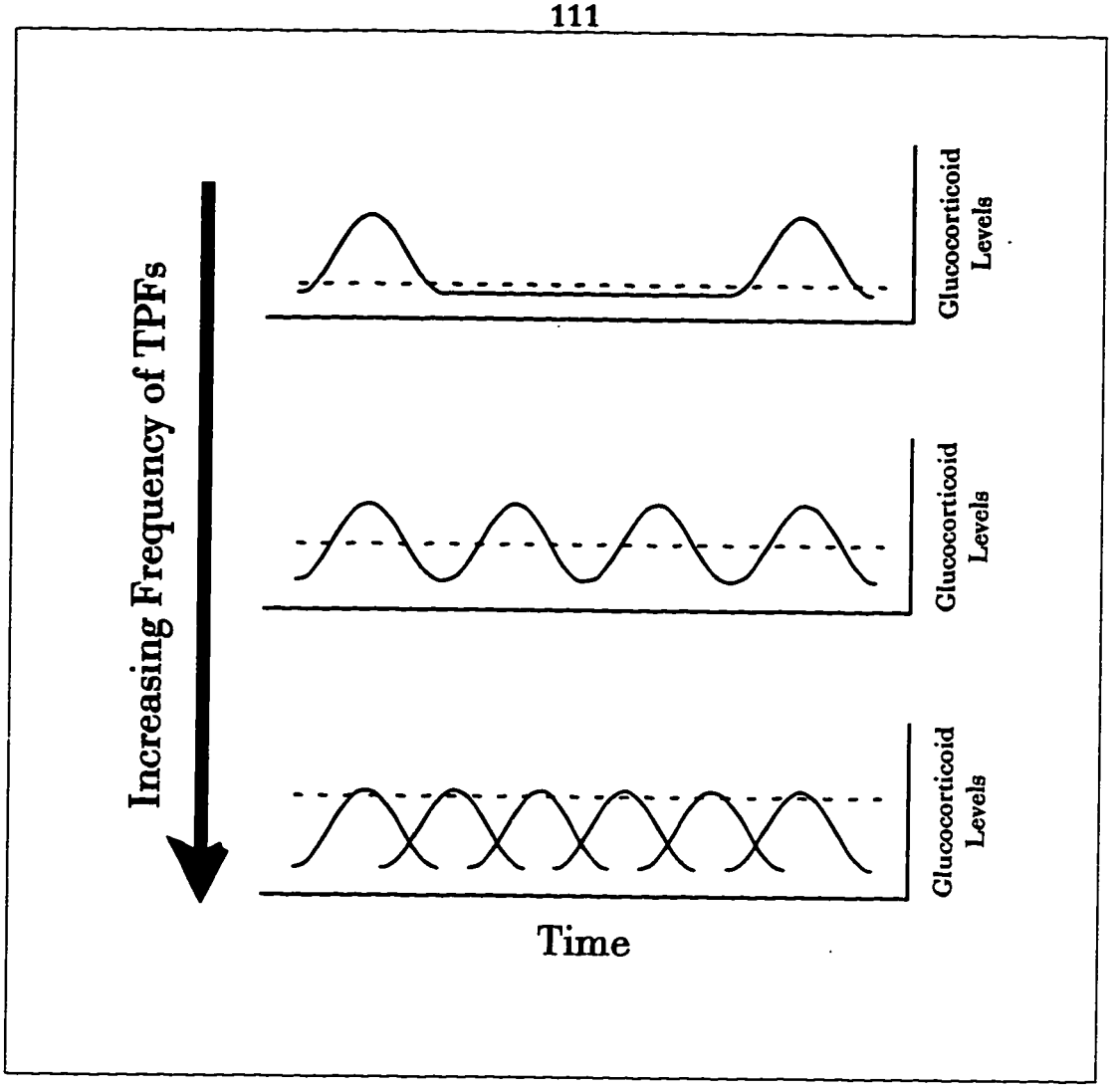


Figure 3.9 - Effect of the frequency of TPFs on basal average glucocorticoid levels in the blood. Solid line curves represent the glucocorticoid response for a single TPF. Dotted line represents the average plasma glucocorticoid level. As the frequency of TPFs increases, so does the average glucocorticoid level. At high frequencies, the average glucocorticoid level is far above basal level.

from the deleterious effects of elevated glucocorticoids. Thus in environments where the predictability is low, the adaptive value of the TPF response will be low since the maintenance of high glucocorticoid levels would drastically reduce the fitness of the animal. As the adaptive value at both high and low ends of the predictability range would be low, the resulting plot of adaptive value versus predictability would result in a large plateau with steep sides at the extremes of predictability (solid line - Figure 3.8). The curve need not me smooth or symmetrical, but should follow the general pattern shown in Figure 3.8. The response to TPFs is very complicated, involving several hormones, a large number of target organs and a complex distribution of glucocorticoid and mineralocorticoid receptors (see Meaney *et al.*, 1996). There will always an energetic cost to maintaining any complex system (Figure 3.8, shaded region). On top of this base line maintenance cost for the TPF response system will be the cost associated with expressing the TES. As the predictability of the environment decreases, the frequency of TPFs increases, and so does the overall cost for required to respond to TPFs. At very low predictabilities, the cost of maintaining the TPF response system will be much higher than in highly predictable environments as there will be far more TPFs.

At the extremes of environmental predictability, the cost of maintaining the TPF response may exceed the adaptive value of the response (Figure 3.8 - areas where solid line drops below the dashed line). Under normal conditions, there

may be an attenuation of the stress response at either end of the predictability scale, particularly at the low predictability end of the relationship.

Though it may be expected that the TPF response will be attenuated in animals that live in either very predictable, or very unpredictable environments, predictability is hardly the only variable determining the physiological state of the animal. Many other factors (such as energy reserves and time of year) will be important in determining the animal's state, and hence its response to TPFs. Figures 3.10-3.12 show the effects of latitude, time of year, and predictability on the adaptive value of the TPF response for a short lived animal. In this example, birds breeding in tropical regions will show a broad but shallow depression in the adaptive value for a TES during the breeding season, and this drop will be well above the critical adaptive value at all times, except in environments which are either very predictable, or very unpredictable (Figure 3.10). At temperate latitudes, there would be a narrower but deeper dip in the adaptive value for a TES during the breeding season, but again it would not be near the critical adaptive value except in environments at the extremes of predictability (Figure 3.11). In the birds breeding at high latitude, the adaptive value will drop below the critical value at all times, regardless of the predictability of the environment (Figure 3.12).

Embryonic Responses to the Unpredictable

Animals use TESs to help increase survivorship in "stressful" situations, but

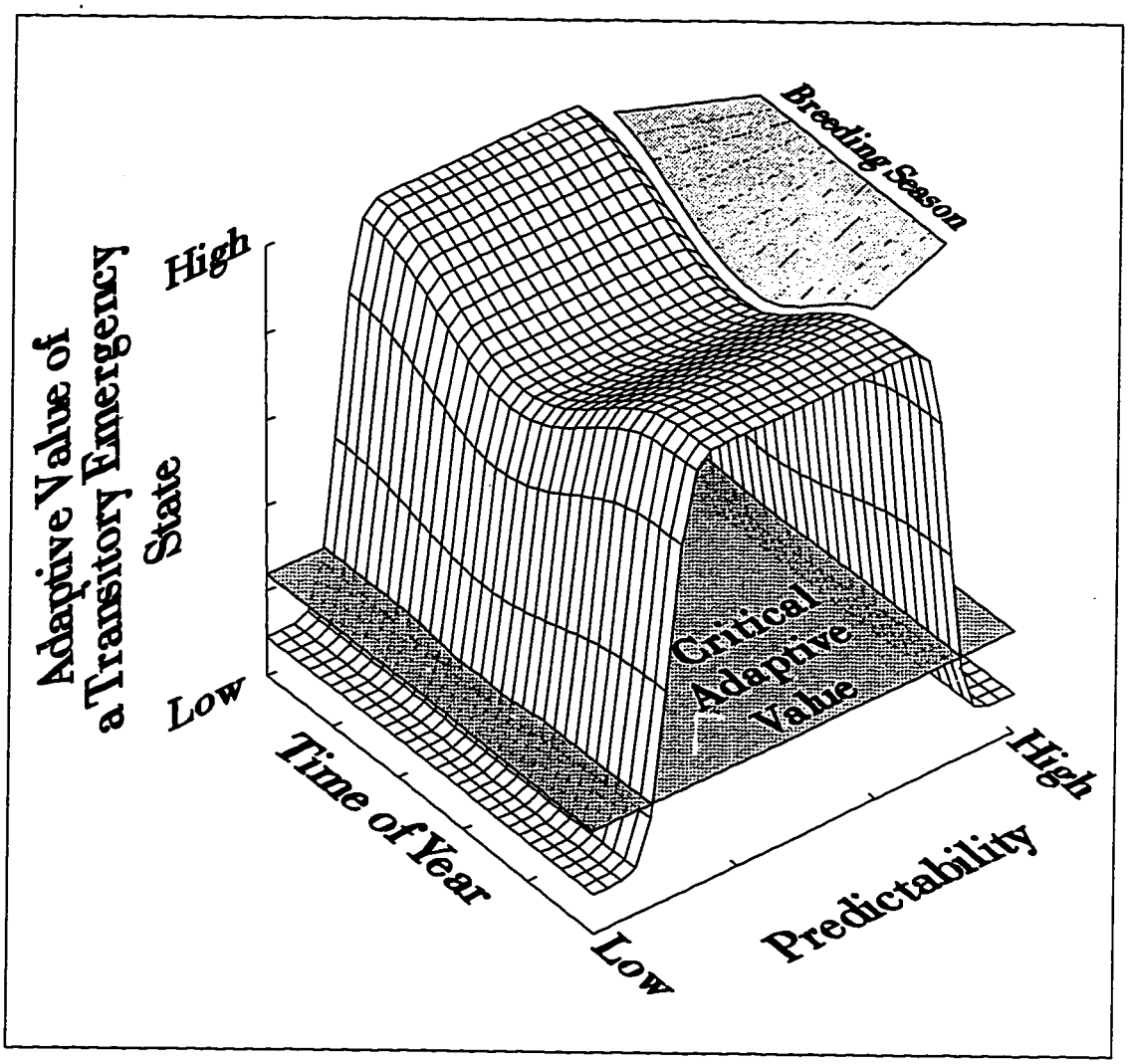


Figure 3.10 - Effect of time of year and predictability on the adaptive value of a transitory emergency state (TES) for a tropical breeding bird. The grey plane represents the critical adaptive value. As the breeding season is long, there is only a small depression in the adaptive value of the TES associated with the breeding season.

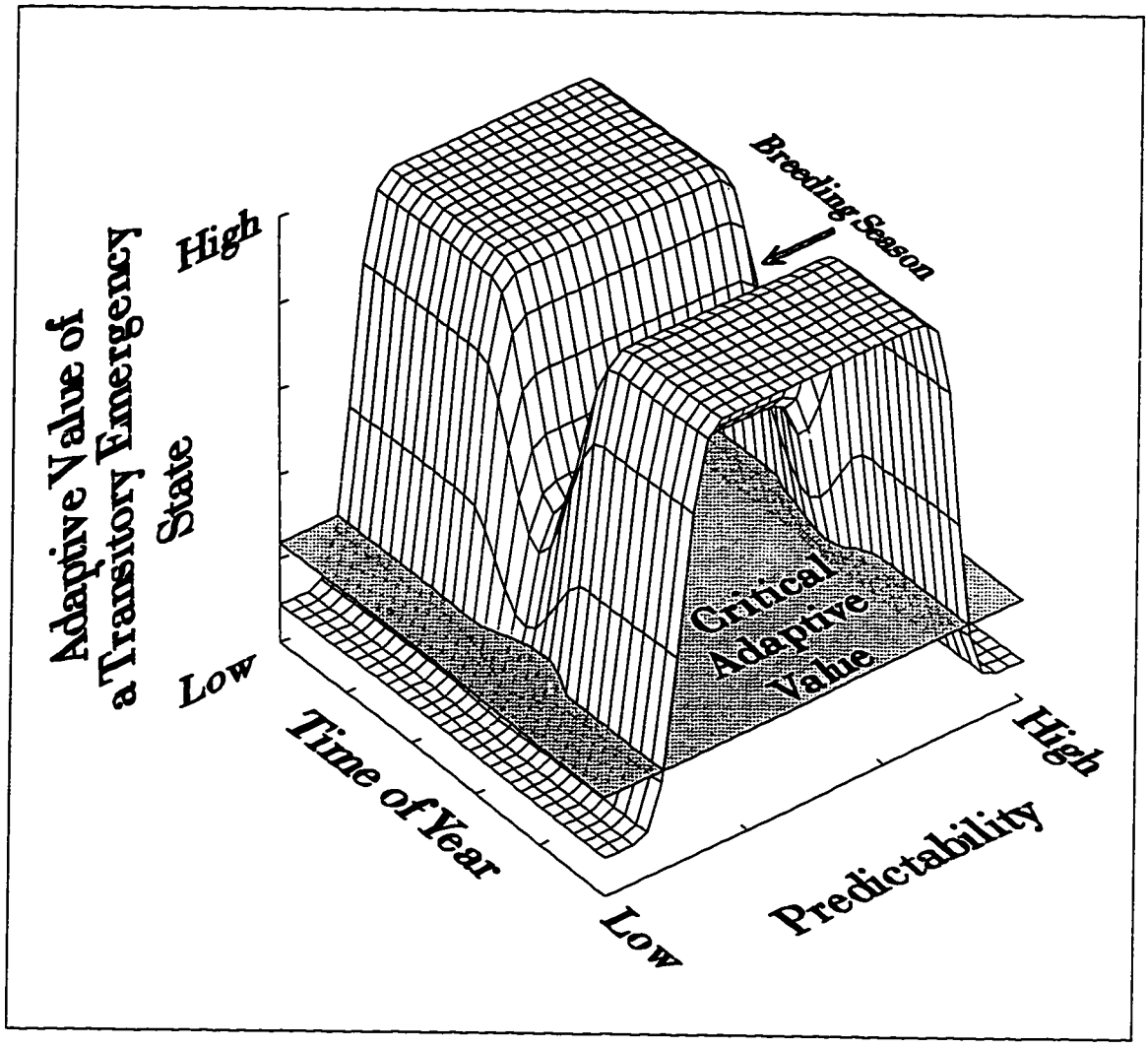


Figure 3.11 - Effect of time of year and predictability on the adaptive value of a transitory emergency state (TES) for a temperate bird. The grey plane represents the critical adaptive value. The breeding season is shorter than in the tropics, but still long enough normally for multiple broods. There is a larger depression in the adaptive value of the TES associated with the breeding season than in the tropics, but it is still normally above the critical adaptive value.

there are some situations where most TESs are not possible. Common laboratory conditions prevent an animal from responding in normal ways to unpredictable events. Animals have not been in captivity long enough to adapt completely to lab conditions, so there has been little or no change in the TPF response resulting from this constraint on their environment. There is however a naturally constrained portion of all animal's life cycles: during embryonic development.

The process of development is a very crucial, and fragile period of an animal's life. Cell division can be rapid, and many tissues must interact at proper times in order for proper development to occur. The release of growth factors and morphogens is important for pattern formation in many tissues and organs. The release of sex steroids is often important in development of primary sex characteristics. During a response to a TPF, glucocorticoids (GCs) become elevated, and these elevated GCs have been show to inhibit many of the above developmental processes. The release and action of many growth factors is inhibited by GCs (Munk *et al.*, 1984), as well as the release of growth hormone(Slotkin *et al.*, 1992). GCs act on the hypothalamus and pituitary gland to inhibit the release of luteinizing hormone and follicle stimulating hormone (Collu *et al.*, 1988). Nerve cells, at least within some regions of the brain, suffer from neurotoxicity from GCs (Virgin *et al.*, 1991). All of these effects could be extremely deleterious during the process of development, potentially causing deviant development of some tissues, and the de-synchronization of the

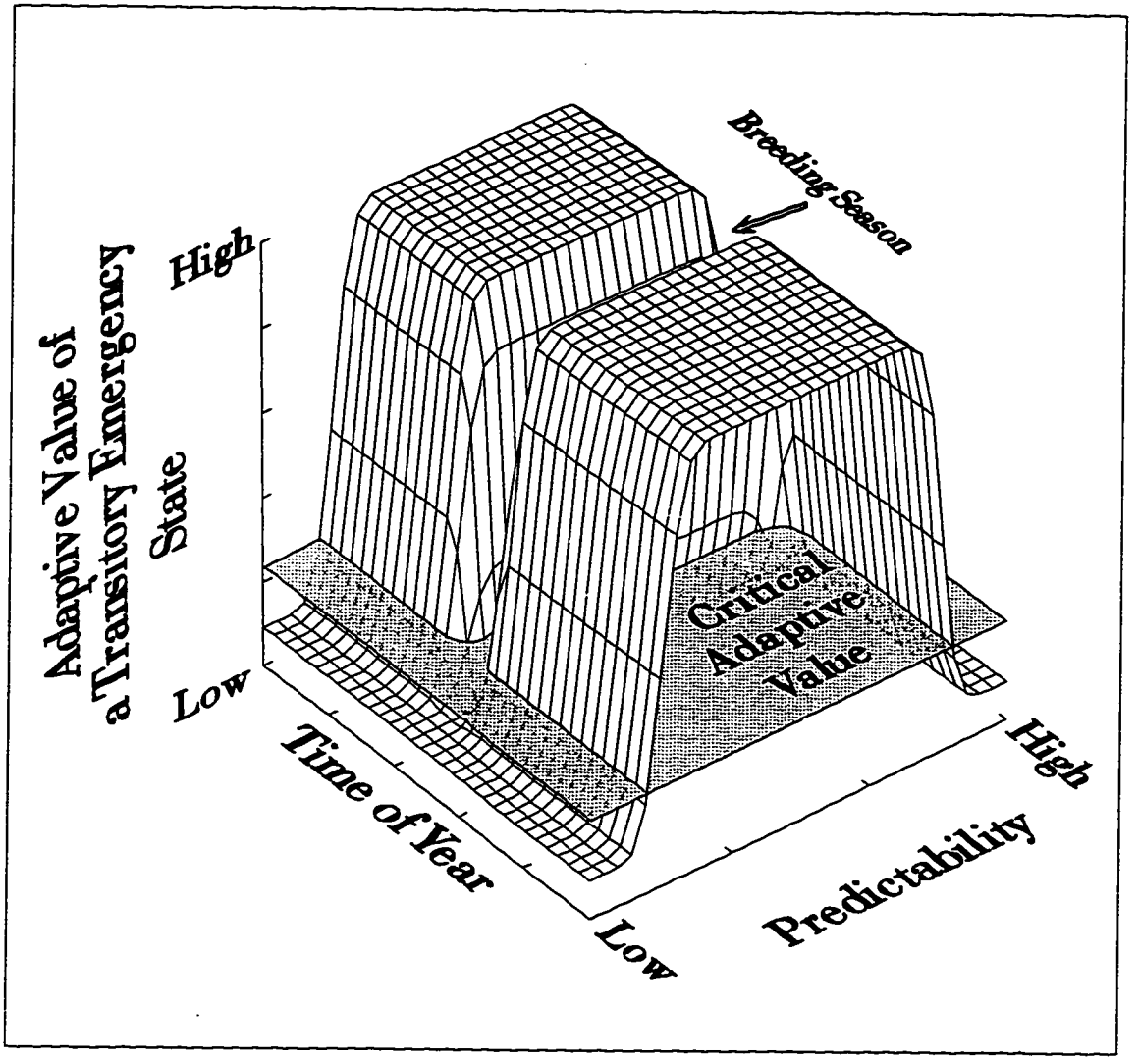


Figure 3.12 - Effect of time of year and predictability on the adaptive value of a transitory emergency state (TES) for a high latitude breeding bird. The grey plane represents the critical adaptive value. As the breeding season is short, there is a large depression in the adaptive value of the TES associated with the breeding season, causing the adaptive value of the TES to be below the critical value throughout the breeding season.

development of dependent tissues.

The developmental processes of an animal follows basically a linear sequence of states from zygote to juvenile, with few possible branching points in the sequence (see figure 3.13). The change of state is driven by a variety of developmental cues: tissue-tissue interactions, growth factors, and gradients of morphogens. In some animals, developmental cues can lead to changes in state leading to either male or female phenotypes, such as in the development of some reptiles (Janzen and Paukstis, 1991) and fishes (Ross, 1990).

The very linear, and generally non-branching nature of the developmental process does not normally allow for the interruptions such as *Transitory Emergency States (TES)* under the influence of perturbation factors, unless if is possible to return to the normal developmental pathway from the TESs. If they cannot return to the normal pathway, the embryo will embark on an abnormal developmental path, leading to a deformed/impaired juvenile, or death. (see bottom path of Figure 3.13).

Since most *transitory emergency states* lead to impaired fitness during hence elevated GCs). Developing embryos are also constrained since they develop with in a "container" (egg, or placenta), so the possible responses of the embryo are also limited in much the same manner as in the laboratory animal, and since many of their response systems may not be functional within the embryo at the time (depending on their stage of development at the time of the TPF). Thus, embryos

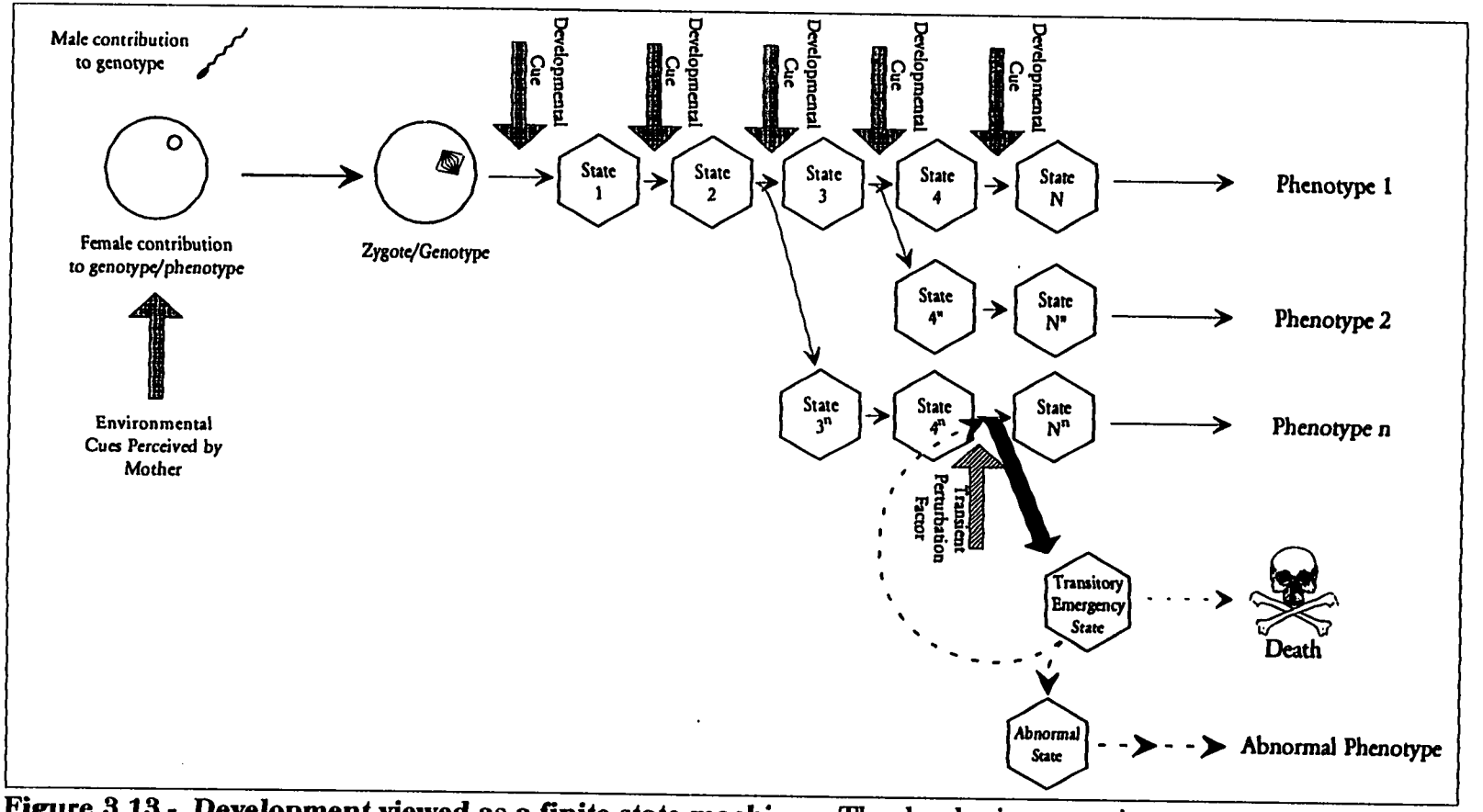


Figure 3.13 - Development viewed as a finite state machine. The developing organism progresses through a mostly linear array of states corresponding to developmental stages. Each stage is subjected to a number of developmental cues, usually from internal sources, but sometimes from the environment. These cues normally lead to the next state in the linear progression, but in some instances, they can lead to branches in the developmental pathway. If the cue is a TPF, the organism will be shunted into a Transitory Emergeny State, and off the normal developmental pathway. Once the TPF is removed, the next state will be determined by the TES, and the cues preceived at that time. In many cases the embryo can not re-enter the normal developmental pathway, leading to death or abnormal development.

are constrained both by their environment, and the need to maintain the normal developmental pathway. We would thus expect to see severe limitations on the TPF response during development (see chapter four for some experimental evidence).

Intensity of TPF and Modulation of the Response to TPFs

As we have seen above, the response to TPFs is dependent on the animal's state at the time of the perturbation. The nature of the TPF is also a factor determining what kind of TES the animal will adopt, and also whether or not the animal will show a response to the TPF. The intensity of a TPF may be important in determining the adaptive value for the TPF response.

A short lived bird which only has a small number of possible broods in a lifetime, and has good energy reserves should normally down regulated its response to TPFs. If however, the TPF is perceived by the animal to be too disruptive, or if the TPF is persistent (such as during a heavy snow fall; see Astheimer et al., 1992) there may be an opposing effect which pushes the adaptive value of the TES back up again (See Figure 3.14). If the short-lived bird is exposed to most TPFs, including capture and mild storms, it will show a muted response to the TPF in order to attempt to continue breeding. If however, the animal is exposed to a severe storm (possibly the result of an early return of winter), the intensity of the TPF may push the adaptive value of the TPF back above the critical adaptive value, triggering the TPF response, causing the animal to

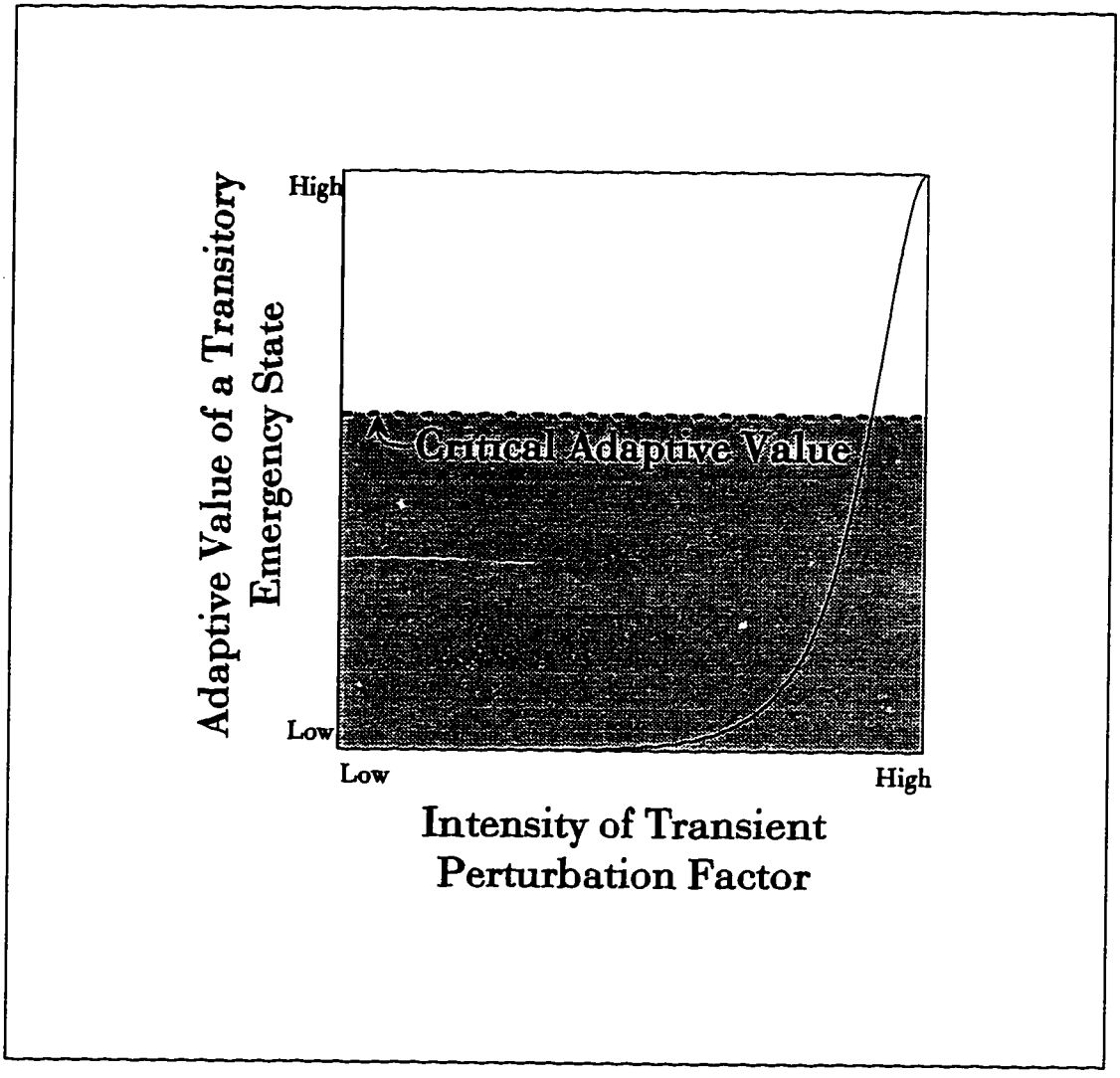


Figure 3.14 - Model of the effect of the intensity of a transient perturbation factor (TPF) on the adaptive value of expressing an transitory emergency state (TES) for an animal under environmental or life history contraints. As the intensity of the TPF becomes high, the adaptive value of the TES increases, and may eventually exceed the critical adaptive value. The critical adaptive value is the dividing line between where the positive aspects of the TES out weigh the negative effects of expressing an TES.

abandon its nest and attempt to save itself. Certainly, if the nest and young are destroyed by the TPF, the animal will no longer be constrained by the need to preserve its offspring.

Conclusions and Implications

The hypotheses generated using this finite state model allow the interpretation of the effects of a number of variables, both excitation variables (such as time of year) and intermediate variables (such as number of broods in a lifetime and energy reserves) on individual responses to TPFs. The dependency of the animal responses to TPFs on the state of the animal (and hence the intermediate variables) and the excitation variables (or environmental cues) indicates that the factors which determine the responses to TPFs may change over time as the animal's state and the environmental cues will change over time. As with the finite state machine model in chapter one, an investigator must know the physiological state of the experimental animal before beginning manipulations, as differences in the starting state of the animal can lead to differences in the responses of the animals.

Chapter IV:

Attenuation of the transitory emergency state (TES) response to transient perturbation factors (TPFs) in a constrained environment: Developing Chicken Embryos

INTRODUCTION

The egg shell protects a developing avian embryo from many noxious stimuli present in the environment. Other transient perturbation factors (TPFs or "Stresses") which effect adults, such as social stress and food limitation, are not applicable to developing embryos. However some factors which have only minor effects on the adult such as humidity, temperature, and oxygen levels all must be kept within relatively narrow limits in order for the embryo to develop properly. The normal incubation temperature of chicken embryos is 38.5°C and incubation temperatures of as little as 2 C° above or 4 C° below 38°C are fatal to the developing embryo (Romanoff *et al.*, 1938; See also figure 4.1). Care by the parental birds can limit the actions of TPFs on the embryos by choosing optimal nest sites and by alter the environment of the egg by warming (and in some cases cooling) the egg. Parental care can not eliminate all possible TPFs. The parent must leave the nest to forage and in species in which only one sex incubates, the eggs may be exposed to reduced temperatures if the weather is cold. Even if the

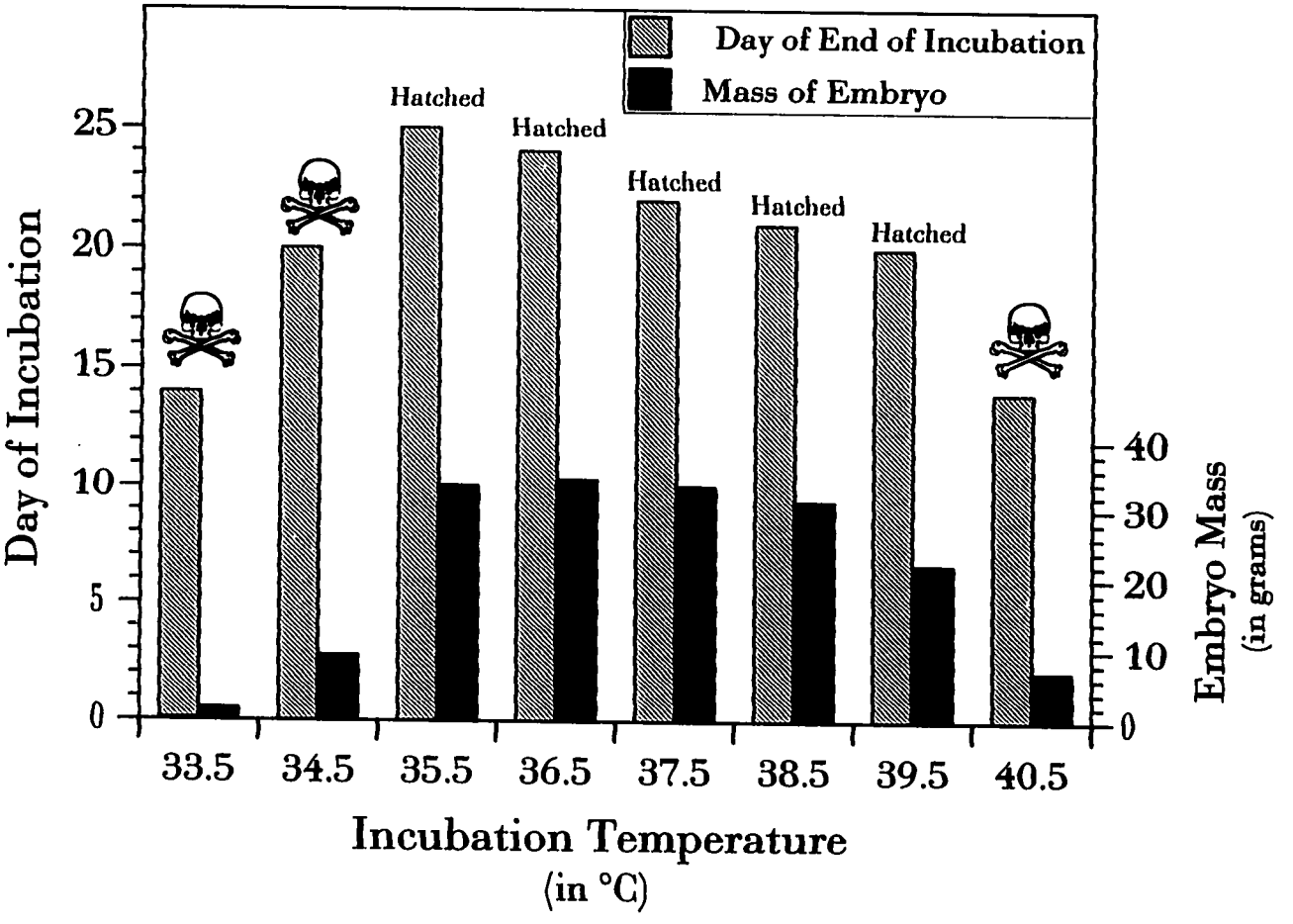


Figure 4.1: Relationship of incubation time and hatching mass to incubation temperature. Drawn from data taken from Romanoff, A.L, (1966). Eggs were incubated at various temperatures, and the date of hatching or death (hatched bars), and embryo mass at the end of incubation (solid bars) were recorded. Bars marked with a skull and crossed bones died before hatching, and those marked with 'Hatched' hatched normally.

parent is present, there is not much a bird such as a chicken can do to reduce the temperature of the egg if the air temperature gets too high.

Embryos are also under a number of constraints absent in adults. As the embryos are locked within a shell, they can not move to a better location to avoid a noxious stimuli, as adults often can. The embryo may not be able to wait for improved conditions, as the developmental can often not be paused, and TPFs such as heat will accelerate many aspects of development and can lead to abnormal development (Primmett *et al.*, 1989). They are also constrained in that they must keep the developmental processes synchronized. Development involves the interactions of many cells and tissues and should the development of different parts of the animal becomes de-synchronized, these tissue interactions may not occur normally (see Gilbert, 1991).

The transitory emergency state (TES) response in adult animals normally leads to elevated levels of glucocorticoid hormones in the blood (*e.g.*, Gorbman, *et al.*, 1983). These steroids cause a number of major problems for the organism if they remain elevated for prolonged times: they can cause neuronal cell death (Virgin et al., 1991) and the death of macrophages (Dohms and Metz, 1991), modify cellular responses to growth factors (Munk *et al.*, 1984), simulate break down of muscle proteins (Dardvet *et al.*, 1995), inhibit wound healing (Munk *et al.*, 1984), cause reduced skin thickness and dermal atrophy (Munk *et al.*, 1984), inhibit the immune response (Wilckens, 1995), and lead to altered blood vessel

function (e.g. DeKloet *et al.*, 1988). All of these effects would be detrimental to an embryo leading to abnormal development and/or death.

While many studies have examined the response of newly hatched and adult birds (e.g. Freeman and Flack, 1980; Ramade and Bayle, 1980; Saterlee, *et al.*, 1980; Freeman, *et al.*, 1984), studies of the embryonic development of adrenal function have been mostly limited to histological studies (Willier, 1930), examinations of levels of steroids in the adrenals (e.g. Kalliecharan and Hall, 1976) and the depletion of steroid precursors from the adrenal (Freeman, 1967). Three studies have examined the developmental time course of the corticosterone release during development of the chick (Wise and Fry, 1973; Kalliecharan and Hall, 1974; Idelman and Marie, 1985). Wise and Fry (1973) also looked at the response by embryos to a TPF, and showed that by day 15 of incubation, the embryos were capable of showing a corticosterone response to a TPF. In this experiment they used a TPF which might mimic a partial predation event: they broke the femur of one leg. Such a TPF is probably not physiologically relevant to the embryo, as they would not be able to survive such a "stress" in the wild.

In this investigation, the effect of temperature change on developing chicken embryos was examined. As the embryos are likely to experience temperature fluctuations during development in the wild, this is a physiologically relevant stimulus. The effects of both cold (22°C and 12°C) and heat (43°C) were examined, as well as the response when the embryos were returned to their normal

incubation temperature after cooling (as would happen when the hen returned to the nest after feeding during cold weather). Corticosterone levels in the plasma of the embryos were measured as an indicator of the degree of embryo's TES response.

METHODS

Study Animal

Cold-fertilized eggs of the domestic chicken, *Gallus gallus domesticus* (Nick Chick variety of the White Leghorn strain), were purchased from H&N International (Redmond, WA). The eggs were incubated at dry bulb temperature of 38°C (± 1°C) and wet bulb temperature of 31°C in a David-Bradley forced air incubator. Eggs were manually turned once each day during incubation. The eggs were candled, at day 12, and the day before treatment, and randomly selected into pools for each time point.

Heating

Eggs incubated at 38°C for 14, 17, 18, 19, or 20 days were then transferred to a second incubator at 43°C. Eggs incubated at 43°C will not hatch (Romanoff, 1960; See also Figure 4.1). Each egg was bled one time.

Cooling

For the cooling experiments, the eggs incubated at 38°C were transferred to cardboard egg trays, then transferred to the appropriate temperature room. For

12°C, the eggs were placed in a 12°C cold room, and for 22°C, the eggs were left at room temperature. Eggs left at either of these temperature will not hatch (Romanoff, 1960; See also Figure 4.1).

Cooling then rewarming:

The eggs were incubated for 19 days at 38°C then transferred into a cold room at 12°C, where they were allowed to remain for 2 hours. The eggs were placed in a cardboard egg tray (with no lid) while they were incubating at 12°C. The eggs were then returned to the 38°C incubator after the 2 hours of cooling.

Egg Temperature Measurement

Measurement of the rate of temperature change within the chicken eggs was determined using a 55 gram unfertilized chicken egg. A hole was drilled in the egg using a 3/32" drill bit, and a micro-thermocouple was inserted into the middle of the egg and the thermocouple was sealed in the hole with melted paraffin. Temperatures were recorded using a Physitemp Electronic thermometer (Model BAT-12, Physitemp Instrument Inc., Clifton, NJ).

Blood Sampling

In order to obtain quick samples, with relatively large volumes of blood, the eggs were cracked open into a 115x50mm preparation dish (08-807, Fisher Scientific) containing a paper towel, and the vitelline blood vessels were exposed where they exit the embryo's body. The vitelline artery was nicked with a pair of

iris scissors, and the blood was collected with heparinized capillary tubes using a mouth pipetting device (see figure 5.2). Blood sampling was completed from 1 to 5 minutes from opening the egg. The capillary tubes were flame sealed at one end, and stored at 4°C until they were centrifuged (2-4 hours). After centrifugation, the plasma was transferred to 500µl plastic microcentrifuge tubes, and stored at -20°C until they were assayed.

Sexing of Embryos:

After the blood sampling was complete and the embryos were euthanized, the embryos were sexed. An incision was made through the body wall from the cloaca to the sternum, and the viscera was pushed aside exposing the gonads. Examination of the gonads allowed positive identification of the sex on and after day 14 of incubation.

Corticosterone Radioimmunoassay:

The assay procedure was a modification (without chromatography) of the methods described by Wingfield and Farner (1975) and Ball and Wingfield (1987). All plasma samples (ranging in volume from 20-100 µl) were equilibrated with 2,000 cpm of tritiated corticosterone (New England Nuclear) and 0.3 ml of distilled water overnight at 4°C. They were then extracted in 5 ml of freshly redistilled dichloromethane. The organic phase was aspirated and dried under a stream of nitrogen at 40°C. Dried extracts were reconstituted in 0.55 ml of

phosphate-buffered saline and 200 µl aliquots taken to duplicate assay tubes and 100 µl to a scintillation vial (with 4.5 ml of scintillation fluid: 4 g Omnifluor, New England Nuclear, in 1 liter of toluene). The cpm of tritium in this vial for each sample provided an estimate of percent recovery following extraction.

The radioimmunoassay utilized an antibody to corticosterone (B21-42, Endocrine Sciences, Tarzana, CA). Of the steroids likely to be encountered in avian blood, progesterone showed 57.8% cross reaction with the antibody. However, progesterone has a low extraction in dichloromethane, and corticosterone is the main steroid assayed in this system. Reconstituted extracts were incubated with 100 µl of antibody (diluted 1/100 from each vial) and 100 µl of tritiated corticosterone (approximately 10,000 cpm) overnight at 4°C. Separation of bound and free cpm was achieved by addition of 0.5 ml of dextran coated charcoal for 12 minutes at 4°C. All samples were then promptly centrifuged at 2,000 rpm for 10 minutes at 4°C in a Beckmann TJ-6 refrigerated centrifuge. Supernatants (containing bound cpm) were decanted into scintillation vials, 4.5 ml of scintillation cocktail added, and equilibrated overnight before counting in a Beckmann LS3500 system (for 10 min or 2% accuracy).

With each assay, two solvent blanks and a standard sample (containing 1,000 pg of corticosterone) were taken through the entire assay procedure as a check on reliability criteria.

Data for the assays were collected from the scintillation counter by an IBM computer and plasma concentrations of corticosterone were calculated using the program in Appendix B.

Statistics:

All statistical analysis of the data was done using the computer program SYSTAT (Wilkinson, 1990). Corticosterone responses over time were analyzed using a one way ANOVA. Post hoc comparisons were made using routines for this purpose built into SYSTAT. Statistics for embryos treated at 38°C on day 18 were done using a Kuskal-Wallis nonparametric ANOVA, as the data variance was heteroscedastic. Pair-wise comparisons between untreated embryos and treated embryos and between males and females were made using an independent Student's t-test.

RESULTS

Changes in Basal Corticosterone Levels with Age of Development.

Basal corticosterone levels showed several changes during the later course of development (See Figure 4.2). Before day 16, basal corticosterone levels were essentially undetectable (<1 ng/ml). Between days 15 and 17, there was a significant rise in the basal corticosterone level ($F=8.820$, $df=1$, $P<0.003$) up to around 7 ng/ml. A second rise in basal corticosterone levels occurred between days 18 and 19, leading to a significant peak (12.4 ng/ml) in basal corticosterone

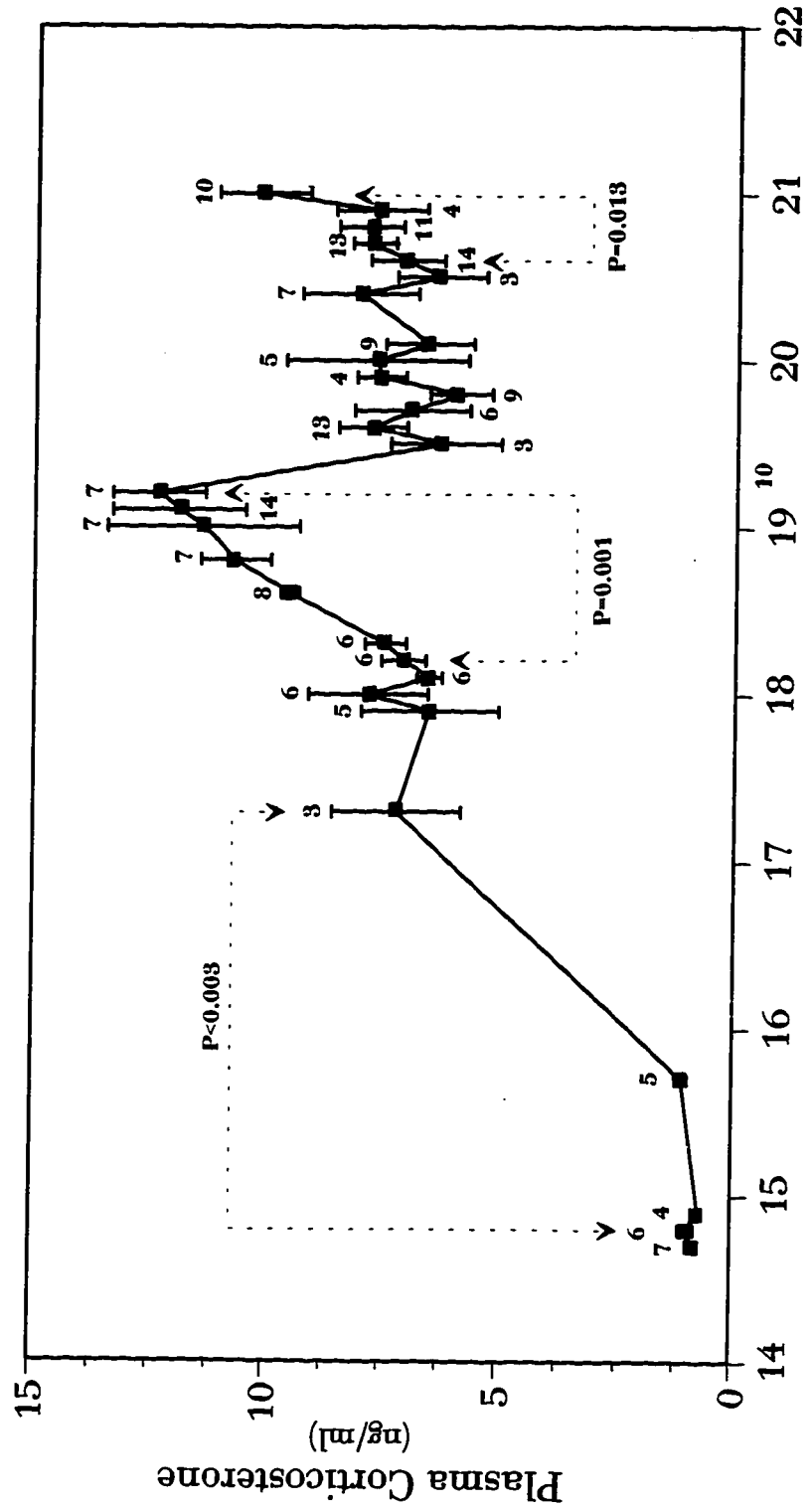


Figure 4.2 - Changes in basal corticosterone with days of incubation. Eggs were incubated continuously at 38°C, and blood samples were taken to determine plasma corticosterone levels at a particular time. Sample sizes are displayed near the time points. Error bars show +/- the standard error of the mean for each data point.

levels (F=10.348, df=1, P=0.002). This peak rapidly tailed off to approximately the day 17 levels(~7 ng/ml) by mid day 19 (F=10.945, df=1, P=0.001 and this level continued till almost day 21. Just prior to day 21 and hatching, there was again a significant increase in basal corticosterone levels (F=6.295, df=1, P=0.013) up to 10 ng/ml.

Temperature Changes in Chicken Eggs.

Heating: Using a 55 gram unfertilized egg, it takes 30 to 40 minutes to reach 43°C when transferred from a 38°C to a 43°C incubator (See figure 4.3). By 20 minutes, the internal egg temperature is nearly 41°C.

Cooling to 22°C: Temperature change from 38°C within the egg to 22°C took 80 minutes (See Figure 4.4). Internal egg temperature were well below normal incubation temperature within 20 minutes of the transfer to 22°C.

Cooling to 12°C: Cooling of the egg to 12°C took 120 minutes (see Figure 4.5). Internal egg temperatures dropped well below normal incubation temperatures within 10 minutes of the transfer.

Rewarming from 12°C to 38°C: Rewarming in the incubator was rapid, taking 50 minutes. Temperature within the egg were in the low end of the incubation range within 30 minutes of the transfer from 12°C to 38°C. (See Figure 4.5).

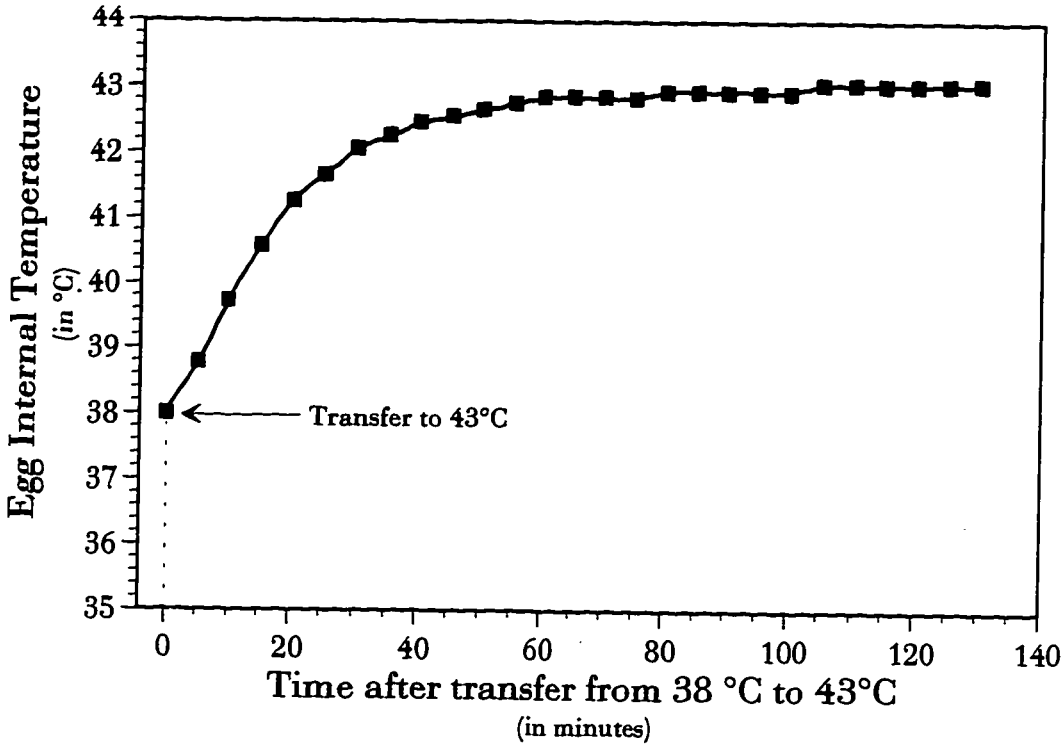


Figure 4.3 - Rate of temperature change in a chicken egg transferred from 38°C to 43°C. Temperature measurements were made using a thermocouple inserted into an unfertillized egg.

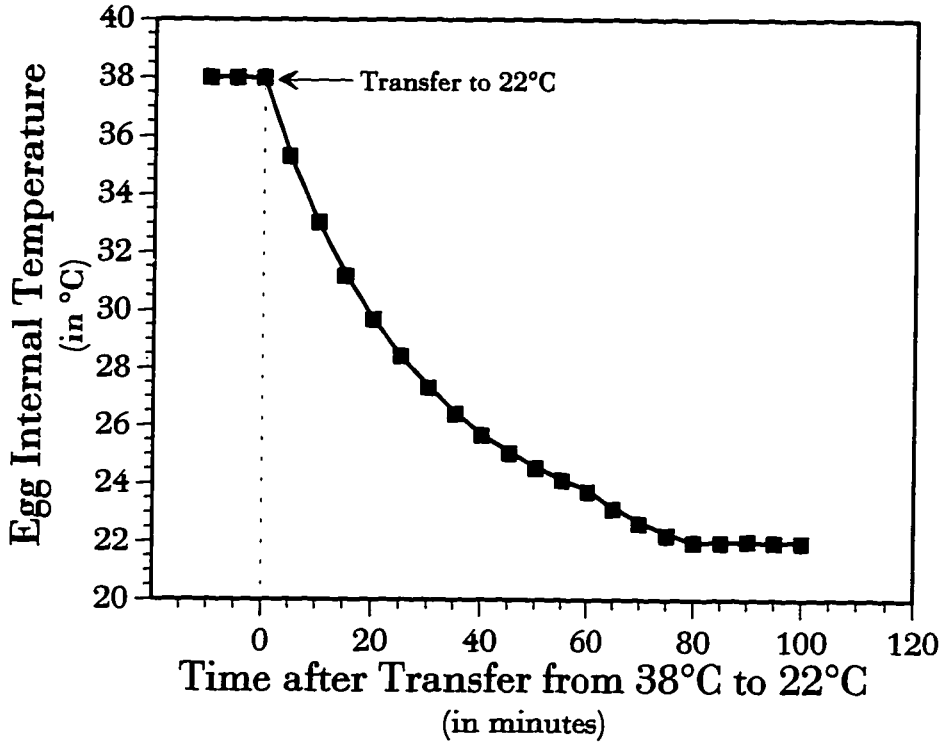


Figure 4.4 - Rate of temperature change in a chicken egg transferred from 38°C to 22°C. Temperature measurements were made using a thermocouple inserted into an unfertillized egg.

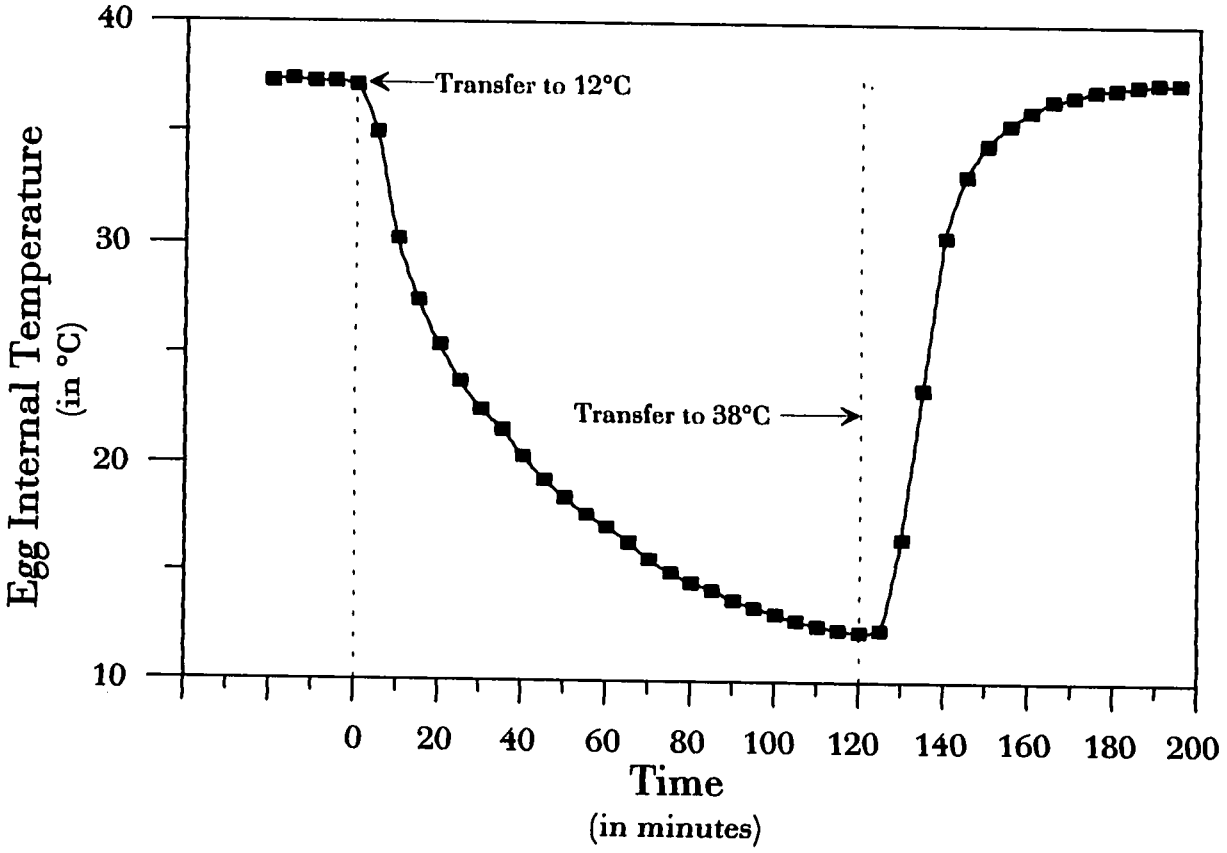


Figure 4.5 - Temperature change within an egg when transfered from 38°C to 12°C, then rewarmed to 38°C. Temperature measurements were made using a thermocouple inserted into an unfertillized egg. Cooling was done in a cold room at 12°C with limited air movement while reheating was done in a forced air incubator at 38°C.

Plasma Corticosterone Levels in response to cooling.

Cooling day 18 embryos from 38°C to 22°C did not produce a significant change in corticosterone levels during the experimental period (Figure 4.6; F=0.289, df=5 P=0.916). The untreated day 18 embryos (left at 38°C), also show no change in corticosterone levels over time (Figure 4.6; H=7.229, df=5, P=0.204)

The untreated day 19 embryos showed a significant decrease in corticosterone levels during the experimental period (Figure 4.7; F=3.937, df=6, P=0.001). This corresponds to the drop in basal corticosterone levels seen above during the 19th day of incubation. Cooling day 19 (See figure 4.7) embryos to either 22°C or 12°C produced no significant increase in plasma corticosterone levels (F=0.206, df=6, P=0.973 and F=0.086, df=5, P=0.994 respectively). Thus day 19 embryos did not show a TES response to cooling at either of these temperatures. Both the embryos cooled to 22°C and the embryos cooled to 12°C were significantly higher than the untreated embryos at 720 minutes of exposure (T=3.316 , df=15, P=0.004, and T=3.993, df=17, P=0.001 respectively). When cooled, the day 19 embryos did *not* show the normal decrease in corticosterone levels.

Plasma corticosterone Levels in response to heating.

Overall Response to heating: Heating chicken embryos to 43°C caused a significant increase in plasma corticosterone levels on days 17, 18. and 20

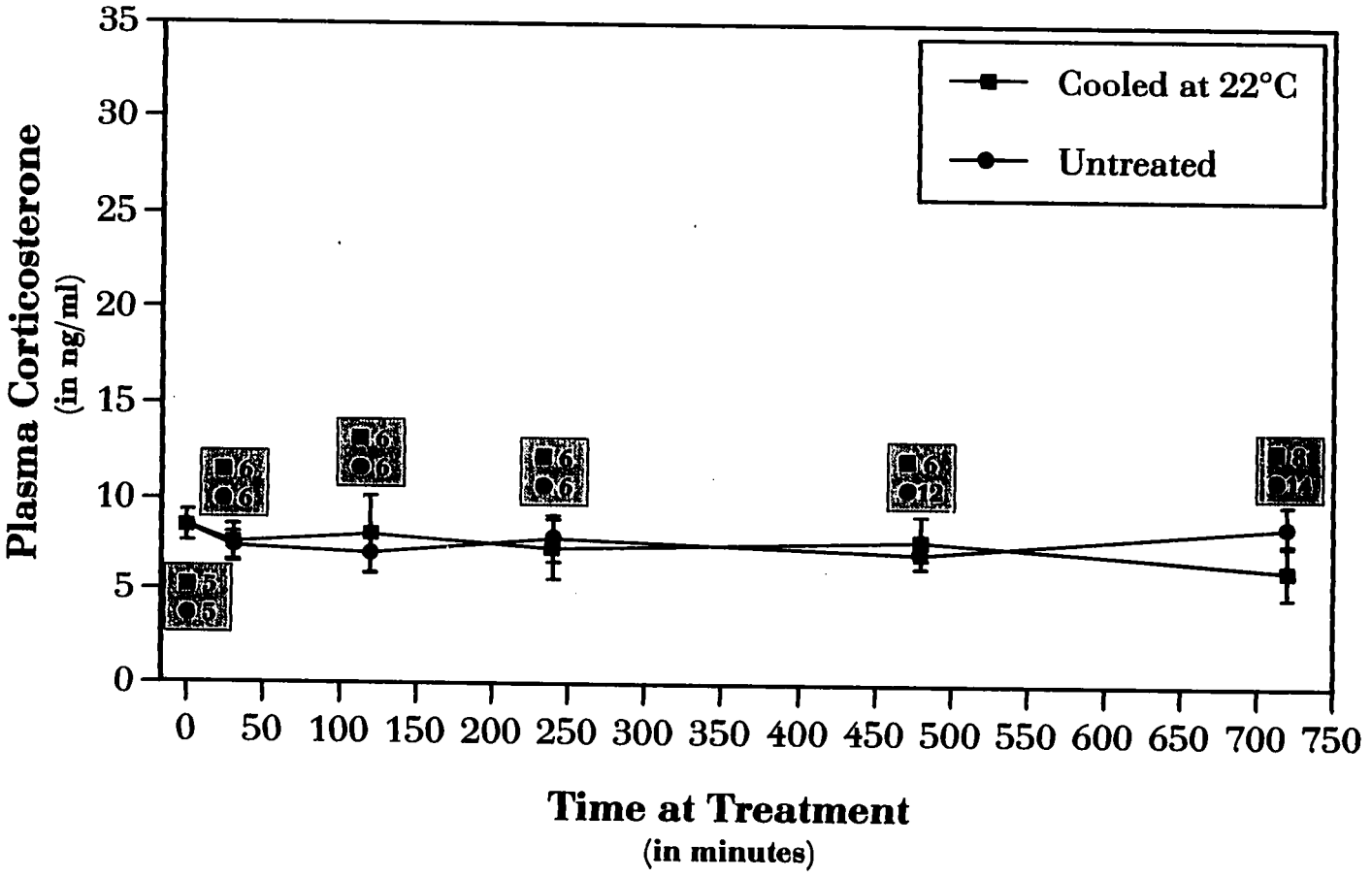


Figure 4.6 - Changes in plasma corticosterone levels of day 18 embryos in response to cooling to 22°C. Day 18 embryos were transferred from the 38°C to the counter top at 22°C (closed squares). Untreated embryos (closed circles) were left in the 38°C incubator throughout the experiment. Sample sizes are shown near the time points with their corresponding symbol. Error bars show +/- the standard error of the mean for each data point.

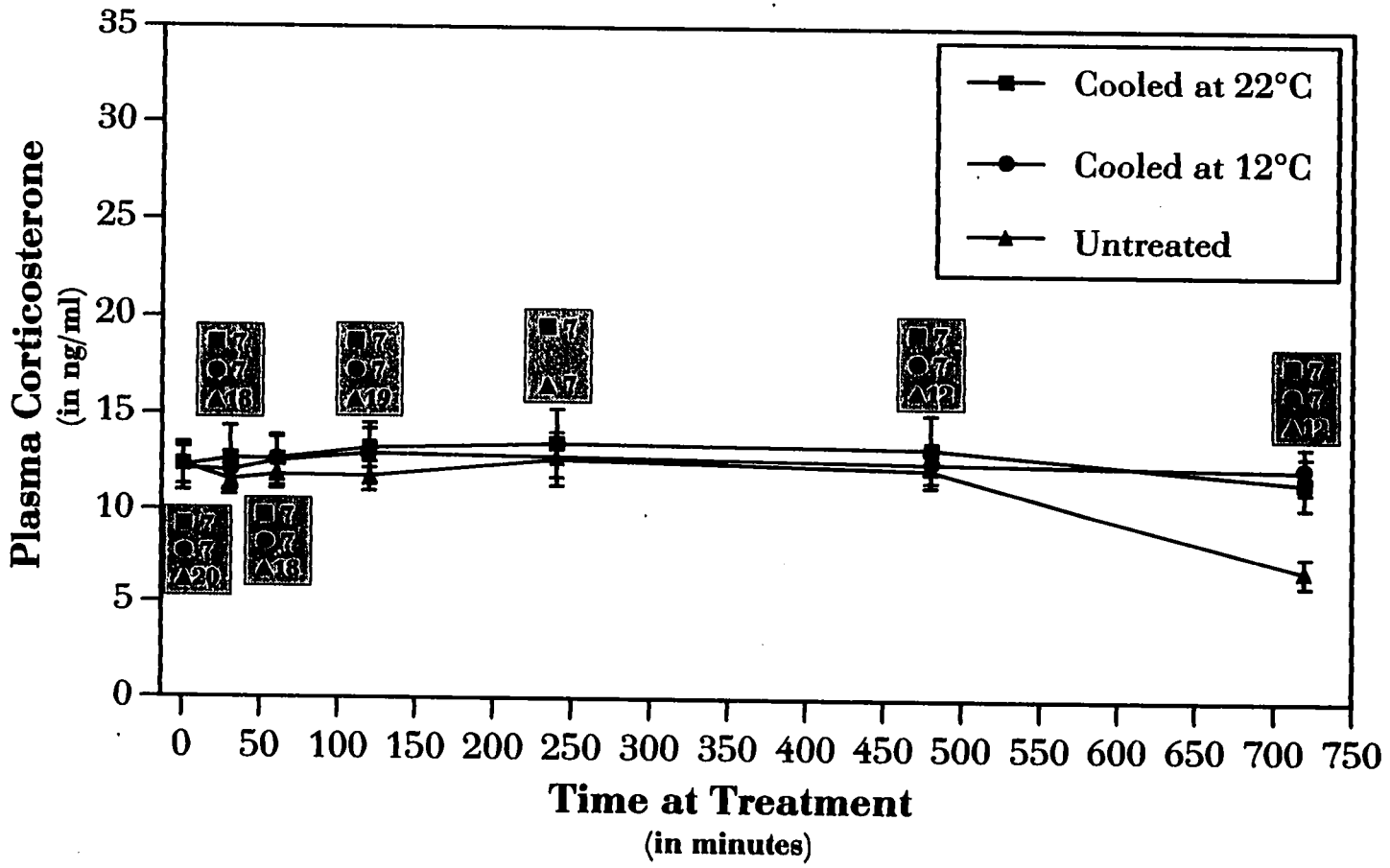


Figure 4.7 - Changes in plasma corticosterone levels of day 19 embryos in response to cooling. Day 19 embryos were transferred from the 38°C to either a 12°C cold room (closed squares), or to the counter top at 22°C (closed circles). Untreated embryos (closed triangles) were left in the 38°C incubator throughout the experiment. Sample sizes are shown near the time points with their corresponding symbol. Error bars show +/- the standard error of the mean for each data point.

(F=5.707, df=5, P=0.001; F=5.295, df=7,P<0.001; F=11.671, df=10, P<0.001 respectively), but not on day 14 or day 19 (F=1.048, df=3, P=3.87; F=0.825, df=4, P=0.524; see figure 4.8). Corticosterone levels in blood samples from day 14 embryos were so low as to be virtually undetectable: most of the samples were below the standard curve minimum detectable dose (<0.715 ng/ml).

Response during the first 2 hours of heating (See Figure 4.9): During the first 2 hours of heat exposure, day 17, 18, and 20 embryos exhibited significant increases in plasma corticosterone levels (F=5.219, df=1, P0.030; F=11.880, df=1, P=0.001; and F=21.691, df=1, P<0.001, respectively). Neither day 14 nor day 19 embryos had significant changes in corticosterone during the first 2 hours (F=2.031, df=1, P=0.165 and F=0.184, df=1, P=0.672). The maximum values for day 17, 18,19, and 20 were not statistically different (F=0.129, df=3, P=0.942), but the basal level for day 19 embryos was higher than day 17, 18, and 20 (F=6.942, df=3, P>0.0004). Basal values for day 17, 18, and 20 embryos were essentially identical (F=0.614, df=2, P=0.544). Both basal and maximal level for day 14 embryos were much lower than all other days (F=205.654, df=1, P<0.001 and F=35.392, df=1, P<0.001 respectively). The maximal levels of corticosterone on days 17, 18, 19, and 20 were not distinguishable from the basal level on day 19 (F=0.231, df=4, P=0.919).

Responses of males and females to heating: Both males and females showed a significant increase in response to heating to 43°C on day 20 (F=24.822,

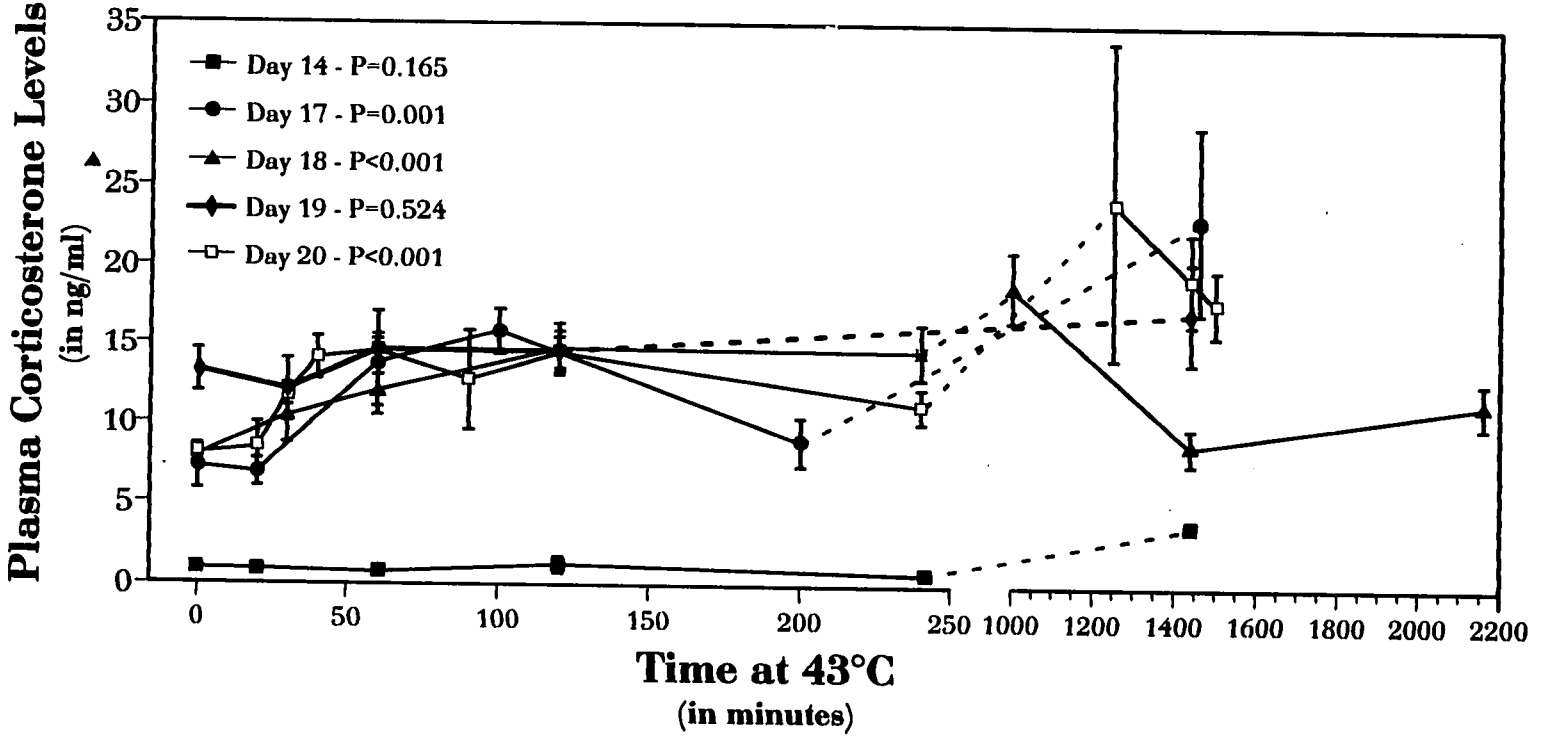


Figure 4.8 - Effect of heating to 43°C on plasma corticosterone levels in developing chick embryos. Embryos of the appropriate age were transferred to 43°C at time zero. Error bars show the +/- the standard error of the mean for each data point. Error bars show +/- the standard error of the mean for each data point.

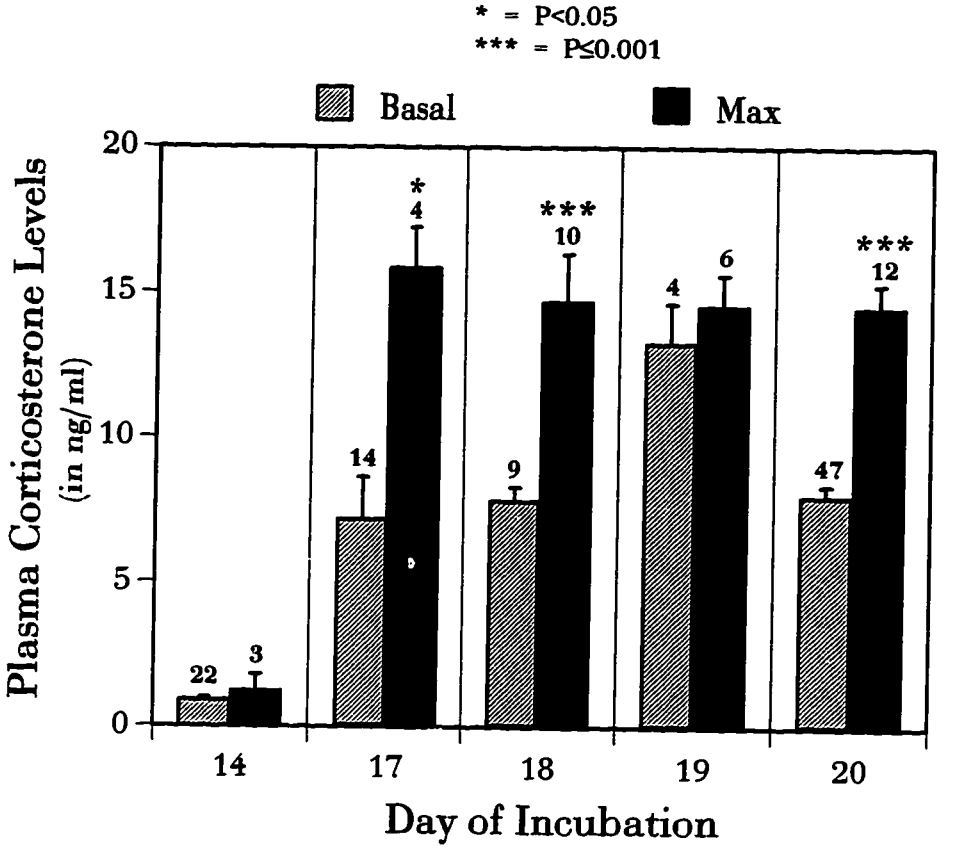


Figure 4.9 - Maximum plasma corticosterone levels in response to the first 2 hours of heating at 43°C. Hatched bars represent corticosterone levels at the beginning of the experiment. Solid bars represent the maximum corticosterone level reached in the first 2 hours of treatment at 43°C. All embryos had previously been incubated at 38°C. Error bars show the standard error of the mean for each bar.

P<0.0001 and F=31.165, P=0.001 respectively), but there was no difference between the response of males and females (T=0.048, df=13, P=0.963 for maximum corticosterone, and T=0.685, df=10, P=0.509 for basal corticosterone levels) (see figure 4.10).

Plasma Corticosterone Levels in Response to Rewarming after Cooling.

Day 19 embryos returned to 38°C, after cooling to 12°C for 2 hours, showed a significant increase in plasma corticosterone levels at 30 minutes after returning to the 38°C incubator (F=7.562, df=1, P=0.011; See Figure 4.11). This increase was very transitory, and the plasma corticosterone levels had returned to basal levels by 60 minutes. Neither embryos left at 12°C (F=0.114, df=3, P=0.951) nor those left at 38°C (F=0.163, df=3, P=0.921) showed any significant change during the 2 hour test period.

DISCUSSION

Basal corticosterone levels in the developing chicken embryo show a number of interesting changes. The capacity of the adrenal glands to secrete corticosterone is extremely limited in embryos before day 15, with plasma corticosterone levels virtually undetectable on days 12 (Wise and Frye, 1973) and 14. By day 17, basal levels of corticosterone are at levels identical to the average on days 18 and 20.

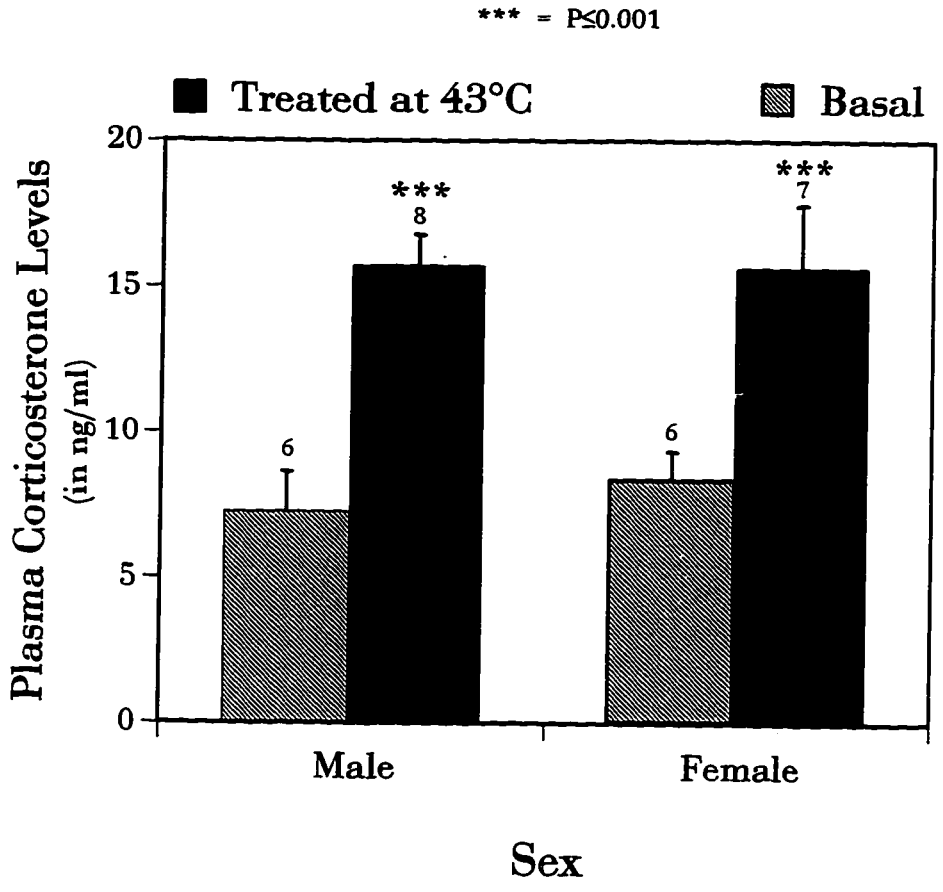


Figure 4.10 - Response of males and females of day 20 chick embryos treatment at 43°C. Hatched bars represent corticosterone levels at the beginning of the experiment. Solid bars represent the maximum corticosterone level reached in the first 2 hours of treatment at 43°C. All embryos had previously been incubated at 38°C. Error bars show the standard error of the mean for each bar.

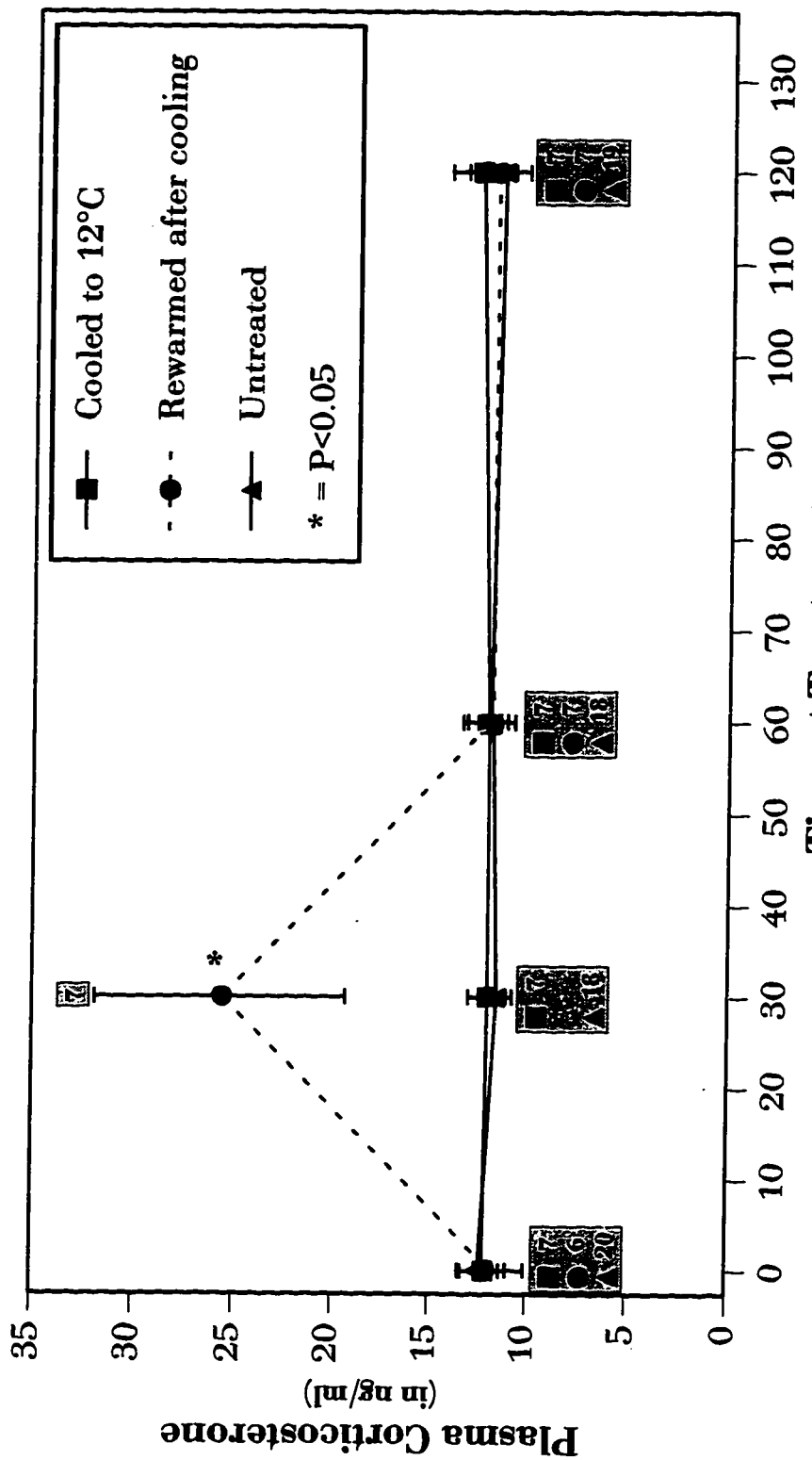


Figure 4.11 - Changes in plasma corticosterone levels of day 19 embryos in response to rewarming after cooling to 12°C. Rewarmed after cooling embryos (dotted line and closed circles) were transferred from a 38°C incubator to a 12°C cold room, then returned to 38°C after 2 hours. Embryos cooled at 12°C (solid squares) were transfered from the 38°C incubator to the 12°C at time zero. Untreated embryos (closed triangles) were left in the 38°C throughout the experiment. Error bars show +/- the standard error of the mean for each data point.

There is major increase in the basal corticosterone levels beginning by mid-day 18 to early day 19 (see Fig. 4.2). Wise and Frye (1973) did not look at day 19 of incubation, so they did not see this peak in basal corticosterone levels. Neither Idelman and Marie (1985) nor Kalliecharan and Hall (1974) found this peak on day 19, but both studies took only on time point per day and could have missed this sharp peak all together. Internal pipping (puncturing of the egg air space by the beak) occurs in mid to late day 19 of incubation (Visschdijk, 1985). Once the embryo punctures the air space, it begins to breath using its lungs. The rise in basal corticosterone levels just precedes internal pipping (and the use of the lungs) and may be analogous to the rise in glucocorticoids in mammals just before birth which stimulates final differentiation of the lung epithelia (Murphy, 1982).

The basal level also begins to climb just before hatching (day 21 of incubation; see Fig. 4.2). This rise begins just prior to the period when the embryo does the external pipping (the beak punctures the shell). After external pipping, respiration in the embryo increases dramatically (Visschdijk, 1985), and the embryo will soon hatch.

Cold appears not to induce a TES response in chicken embryos (see Figs. 4.6 and 4.7). Development at the reduced temperatures of 22°C and 12°C is retard and inhibits normal developmental changes in basal corticosterone levels. (see figure 4.7).

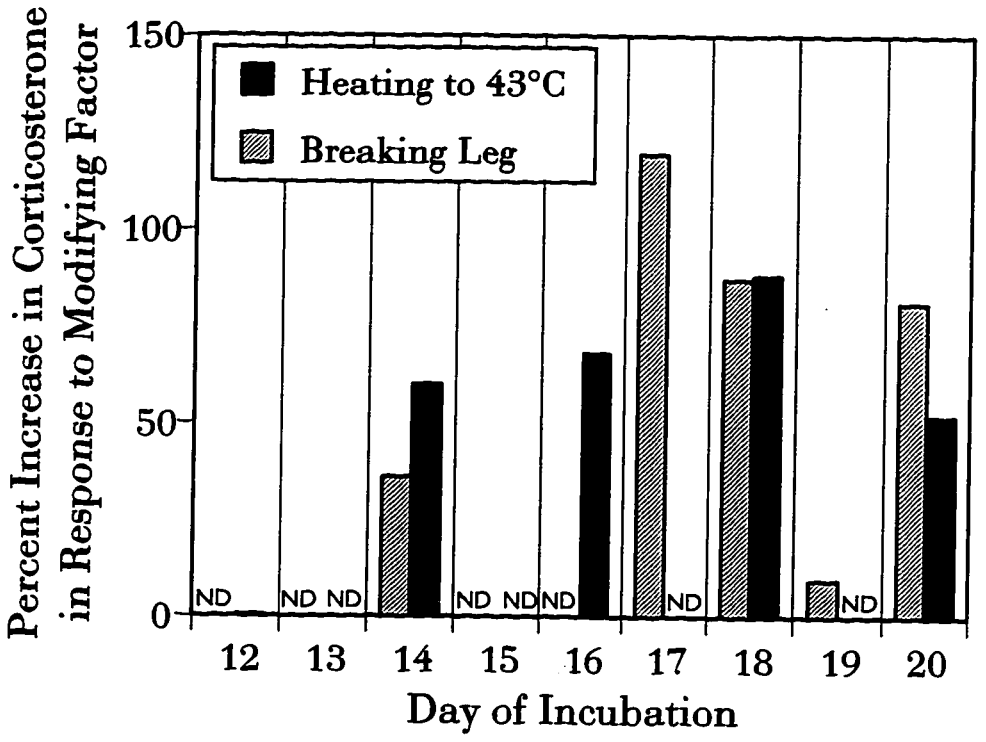


Figure 4.12: Comparison of 2 Label Modifying Factors (Stressors) on increases in plasma corticosterone levels. Cross-hatched bars represent data from Wise and Frye (1973) where the LMF was breaking of the femur. Solid bars represent data from embryos exposed to 43°C for 2 hours.

Heat induces a TES response in day 17, 18, 20, and 21 embryos. Though these embryos show a significant response to the heating, maximum corticosterone level is restricted and is identical to the normal basal level for day 19 embryos. Thus the glucocorticoid aspect of the TES response in chicken embryos seems to be limited within the normal range of basal corticosterone levels, and does not obtain the high, potentially damaging levels show in adult chicken TES responses.

Under the conditions of this experiment, it takes nearly 2 hours for an egg to cool from 38°C to 12°C (see figure 4.5). It is not clear whether a normally developing embryo would cool at a faster or slower rate, since cooling would be promoted by the circulation of blood, but reduced by the heat produced by the embryos at later stages of development (Romanoff, 1941). The embryos in the cooling-reheating experiment were probably down to 12°C by the time they were returned to 38°C or nearly so. During cooling to 12°C, embryos do not show a corticosterone response (see figure 4.7), but when they are returned to 38°C, there is a rapid peak in corticosterone levels by 30 minutes (see Fig. 4.11). This peak is very short lived and the embryos are back to basal levels by 60 minutes after transfer to 38°C. Normally, a TES response does not show this short time course (*e.g.*, Wingfield *et al.*, 1992; Smith *et al.*, 1994). This short lived peak may be the result of changes in circulation in the animal on cooling. If blood flow to the adrenal glands was restricted, and the production of corticosterone continued, you would see an increase in plasma corticosterone levels when the blood flow to the

adrenals returned to normal (*i.e.*, on warming). It is doubtful that this short lived increase in circulating corticosterone is physiologically relevant, since it generally takes on the order of hours for genomic receptors to show a response (O'Malley and Tsai,1992), and the embryo would already be back to its normal temperature by that time.

It seems that for chick embryos, cold does not act as a TPF as it does not elicit a corticosterone response. Moderate heating does elicit a response, but this response is within the normal range of developmental release of corticosterone in developing chickens. The modulation of the adrenocortical response to TPFs is exactly what would be expected in a constrained environment (see chapter three). This limiting of corticosterone levels when high levels might be detrimental is also seen is some birds during the breeding season (Wingfield *et al.* 1994a; Wingfield *et al.*, 1994b).

Chapter V:

Responses to Transient Perturbation Factors in Environments at the Extremes of Predictability: Fishes in the Intertidal Zone and in Caves.

Introduction

The bony fishes (Osteichthyes) comprise the most diverse group of vertebrates, and live in a wide variety of aquatic habitats and even some semi-terrestrial habitats (i.e. Mudskippers: Gobiidae). Some of these habitats, such as in caves, desert springs, Antarctic waters and the deep sea, experience long periods of very stable conditions. These stable environments have generally high levels of predictability, due to constancy.

Other aquatic habitats are not so predictable, and are far more at the whim of the weather. Some fishes, such as the lungfishes *Lepidosiren* (Lepidosirenidae) and *Protopterus* (Protopteridae) and many species of South American and African killifishes (Cyprinodontidae), live in areas that dry up for large portions of the year. Possibly the most chaotic and disturbed environment of all is the intertidal zone (Gibson, 1993). In the intertidal zone, large waves generated by storms can strip animals and plants from their hiding places and smash them on the rock or toss them on the beach. Boulders and logs can be dragged around the intertidal zone, scraping regions free of macro-life. In addition to unpredictable wave action, they are also subject to the rhythm of the tides. During the spring tides (as

opposed to the neap tides), intertidal animals are left behind by the receding waters, often for periods of several hours. Whether in pools or left high and dry, they are exposed to large changes in their environment. The sun can rapidly heat up small pools in the summer causing oxygen concentrations to plummet, and exposed soft-bodied animals are promptly desiccated. Heat and desiccation can trigger the release of noxious secondary metabolites from algae and sponges into tidepools. During the winter, freezing temperatures and snow can kill exposed animals and plants. Rain can also be a major problem, particularly in the Pacific Northwest as rain can dilute small pools, and run-off streams that flow into the intertidal zone can dilute even large pools. The intertidal zone is a harsh and unforgiving environment. (see Ambler and Chapman, 1950; Morris and Taylor, 1983; Daniel and Boyden, 1974; Truchot and Duhamel-Jouve, 1980)

While the rhythm of the tides is generally predictable, wave action and conditions on emersion can be quite unpredictable. A subtidal animal placed in the intertidal zone will normally succumb rapidly in the severe conditions (Gibson, 1993). Obviously, intertidal animals are adapted to these severe conditions, but even they can experience conditions which they are either quantitatively, or qualitatively not able to cope with, and would thus constitute *transient perturbation factors* (TPFs). Non-intertidal fishes, such as salmonids and pleuronectids show large glucocorticoid increases in response to TPFs (Horn and Gibson, 1988). If an intertidal fish were to express such large glucocorticoid responses to TPFs during

the spring tides, this could lead to greatly increased average glucocorticoid levels (see chapter three). Such chronically elevated levels of glucocorticoids could lead to serious problems for the animal.

In this study, I examined fishes from environments the extremes of the predictability range. *Ascelichthys rhodorus* (Rosy-lipped Sculpin: Cottidae) and *Xiphister atropurpureus* (Black Prickleback: Stichaeidae) live almost exclusively in the intertidal zone, and thus are adapted to a low predictability environment. In these fishes, I expected to find the stress response attenuated, to prevent chronically elevated cortisol levels due to frequent TPF events. These intertidal fishes were compared to three subtidal fishes: *Enophrys bison* (Buffalo Sculpin: Cottidae), *Lepidopsetta bilineata* (Rock Sole: Pleuronectidae), and *Pleuronichthys coenosus* (C-O Sole: Pleuronectidae). At the other end of the predictability range I studied *Astyanax mexicanus* (Blind Cave Tetra: Characidae) which is found in caves in Central America and Southern Mexico that have extremely small fluctuations in temperature and food availability throughout most of the year. *Astyanax* was used to test whether the glucocorticoid response to TPFs in a very predictable environment would be attenuated, since it may not be required in a predictable environment.

Methods

Study Animals:

Wild specimens of the intertidal fishes *Ascelichthys rhodorus* (Rosy-Lip Sculpin) and *Xiphister atropurpureus* (Black Prickleback) were captured on the Olympic Peninsula in the Straits of Juan de Fuca just west of the town of Joyce, Washington (long. 124°0'0" W; lat. 48°18'33" N; See Figure 5.1). Specimens were captured during the years 1986-1994. *Ascellichthys* was caught using bare hands, and *Xiphister* was caught using either bare hands or with the assistance of a net or nylon mesh bag.

Wild specimens of the subtidal fishes were caught using SCUBA and hand nets. *Enophys bison* (Buffalo Sculpin) were caught at the Edmonds Oil Dock in Edmonds, Washington (long 122°23'29"W; lat 47°48'17"N), just off shore of Marina Beach, and *Lepidopsetta bilineata* (Rock Sole) was caught both at the Edmonds Oil Dock, and at Richmond Beach County Park (long 122°23'2"W; lat 47°45'53"N; See Figure 5.1).

Specimens of tank raised *Astyanax mexicanus* (Blind Cave Tetra) were purchased from a local retail aquarium store, and housed in bare-bottomed 38 or 57 liter aquaria. The aquaria were maintained in a constant temperature environmental chamber at 22°C, with constant dim light. The aquaria were filtered using air driven sponge filters.

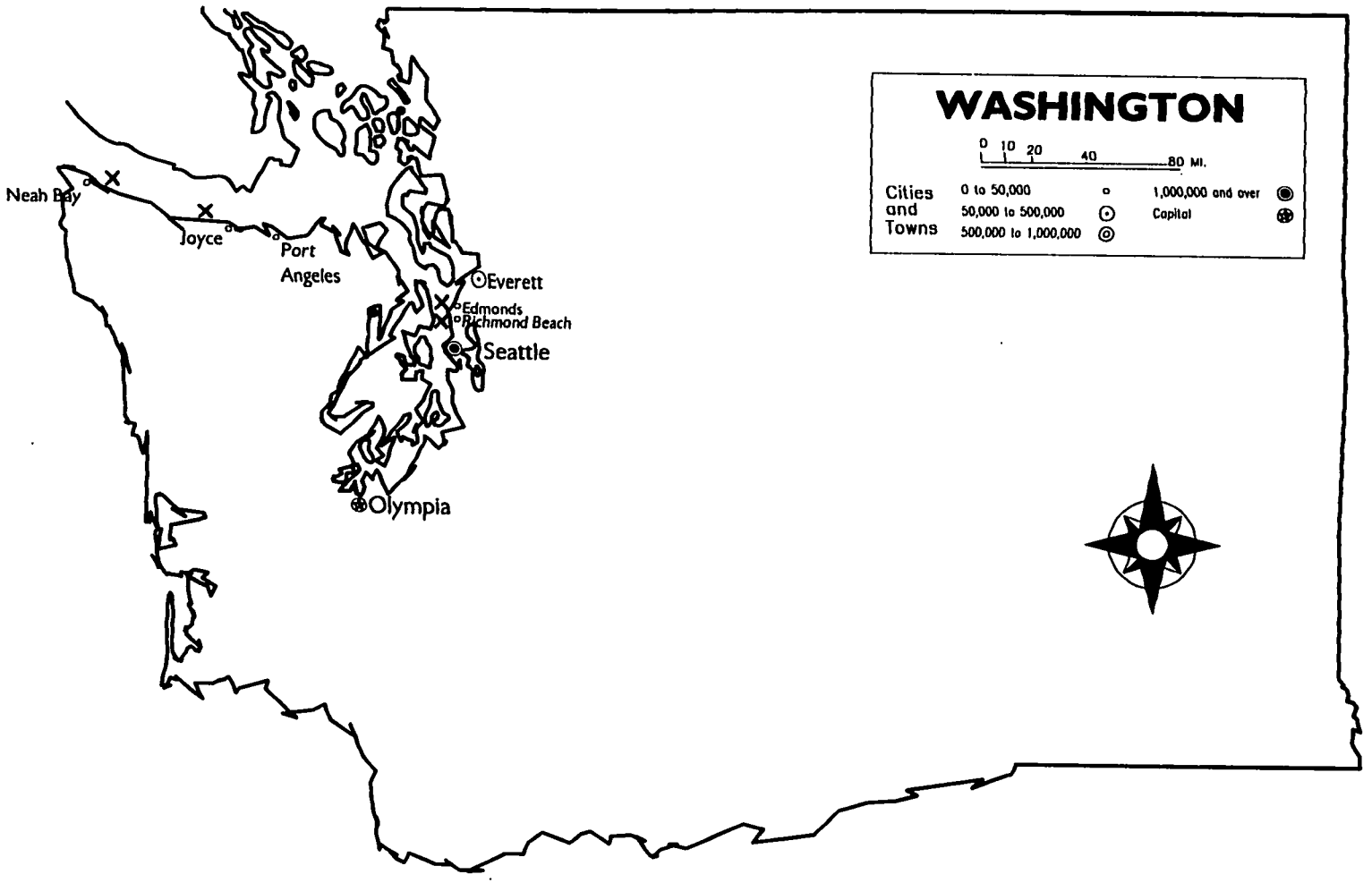


Figure 5.1 - Map of Washington showing study sites. Intertidal fishes were collected near Joyce and Neah Bay, and the subtidal fishes were collected at Edmonds and Richmond Beach.

Blood Sampling:

Bleeding Board: Conditions in the intertidal zone in Washington can be rather wet and windy, particularly during the winter and spring, so steps had to be taken to keep the bleeding equipment in place. A bleeding board was constructed (see figures 5.2, 5.3, and 5.4). The board (Figure 5.3) was constructed of coated 3/4" plywood, with 3 threaded inserts for connecting the elastic hold-down strap and the light arm to the board. The board was cut to fit inside of the backpack used on collecting trips. Cotton applicators were stored in a plastic box, and this box, along with the tubes of capillaries was placed under the elastic hold down strap. The elastic hold down strap also constrained the 27 gauge needles, and forceps to prevent them from blowing away or from being lost down holes between the rocks in the intertidal zone. The light arm (Figure 5.4) was used to illuminate the gill chamber, and greatly facilitated bleeding of the fish, particularly at night.

Mouth pipettor for blood: All blood samples from the fishes were taken with the aid of a mouth pipettor (see figure 5.5). This was necessary since the volume of blood in a sample was often low, and mucus from the gills tended to plug the capillary tubes.

Blood Sample Holder Box: As there is always the danger of falling when walking on slippery intertidal rocks, a method was devised to protect the blood samples from impact damage. A block of foam rubber was cut to shape to fit into a 1 pint freezer storage box, and a series of holes were made in it with a heated

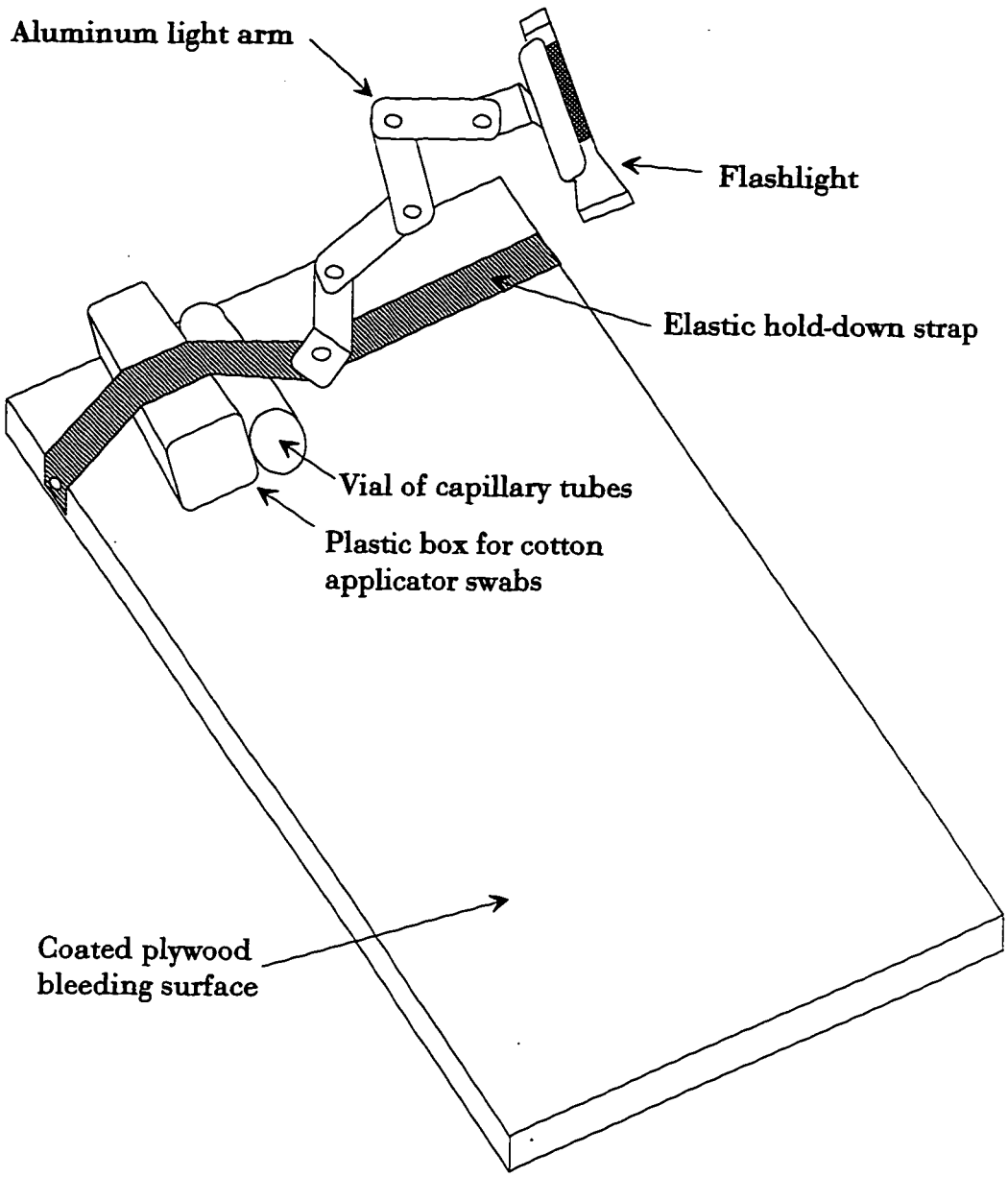


Figure 5.2 - Diagram of assembled fish bleeding board. The elastic hold-down strap keeps the swab box and hypodermic needles from blowing away during windy conditions.

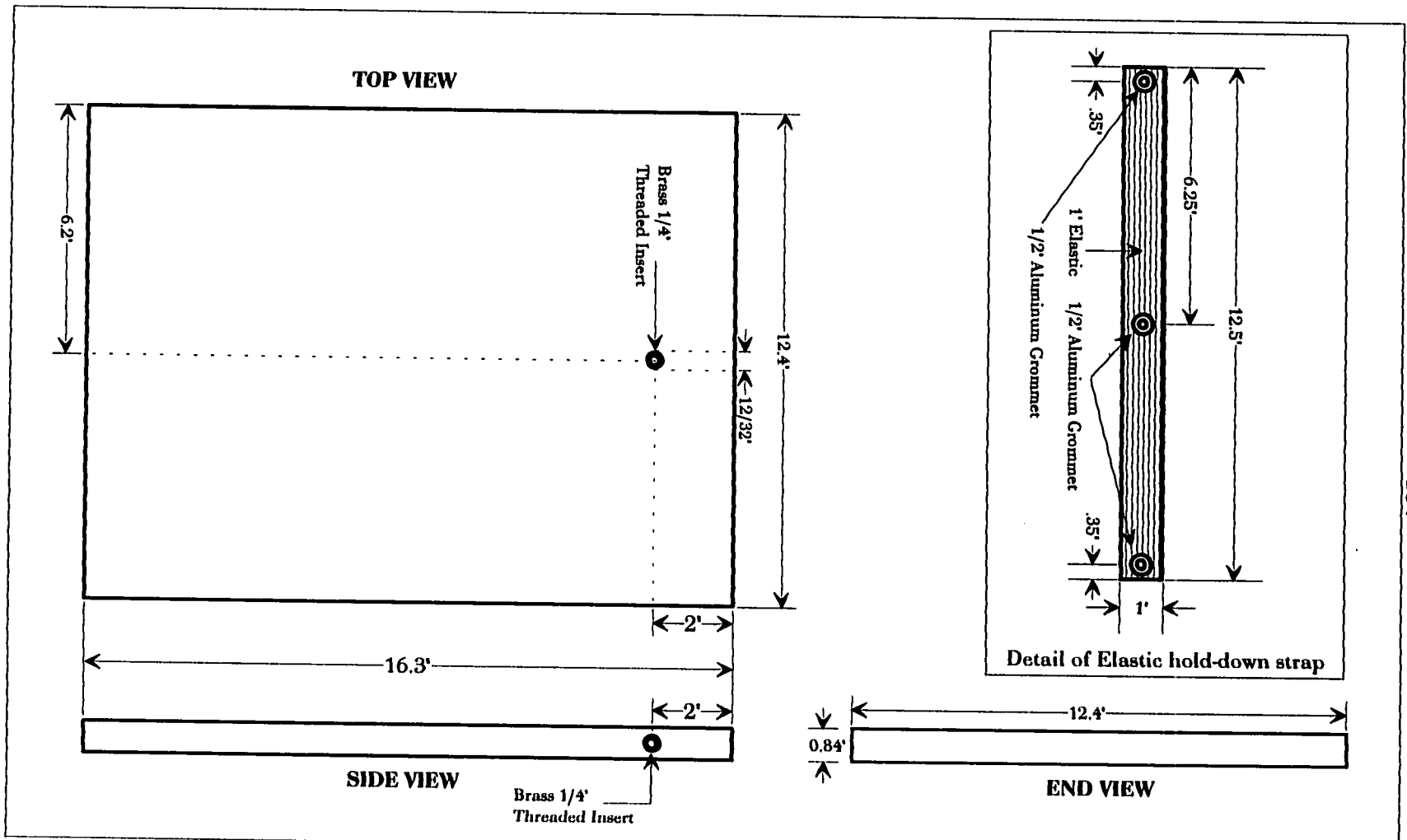


Figure 5.3 - Diagram of Fish bleeding board. The board was cut from a sheet of 3/4' veneered plywood, and had 3 1/4'x20 Brass threaded insert added to postition the hold down strap. The hold-down strap, made from a strip of 1' wide elastic with 1/2' aluminum grommets, was held in place by 1/4'x20x3/4' stainless steel machine screws. The center screw was also used to hold the collapsable light holder. The elastic strap held down instruments which might be lost in the tide pools or blown away by the wind.

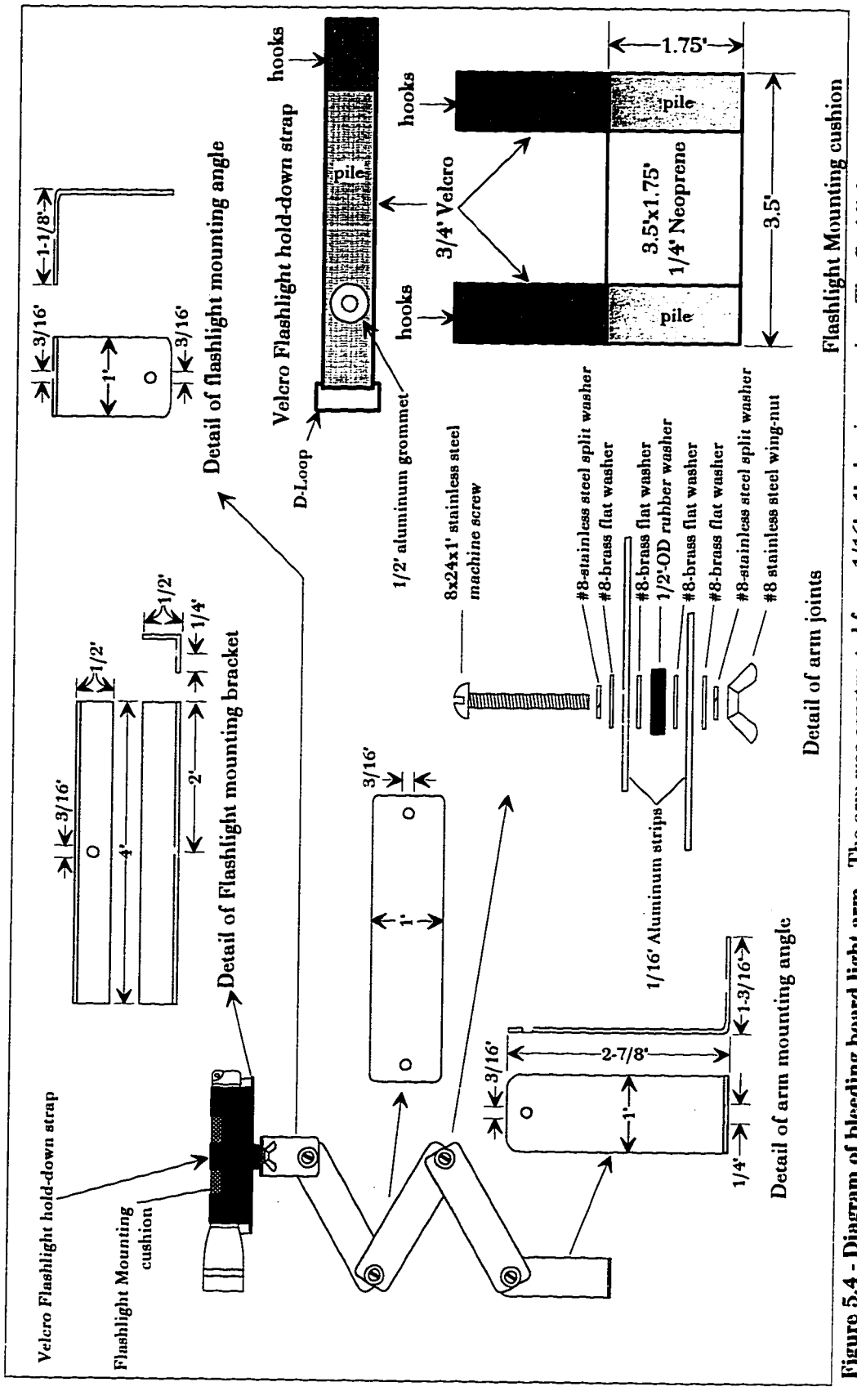


Figure 5.4 - Diagram of bleeding board light arm. The arm was constructed from 1/16' x1' aluminum strips. The flashlight was surrounded by a cussion of neoprene, held in place by 3/4' velcro. The cushioned flashligh was held firmly to the mounting bracket by a velcro strap

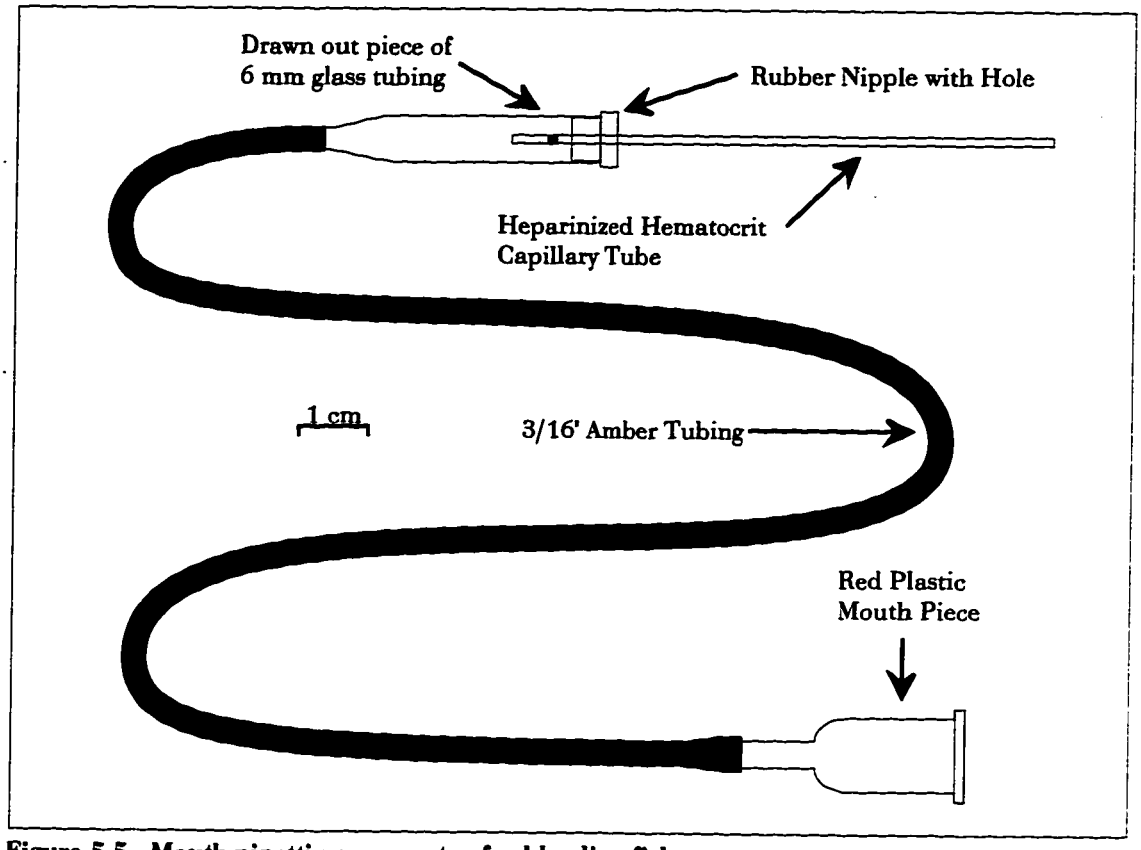


Figure 5.5 - Mouth pipetting apparatus for bleeding fish.

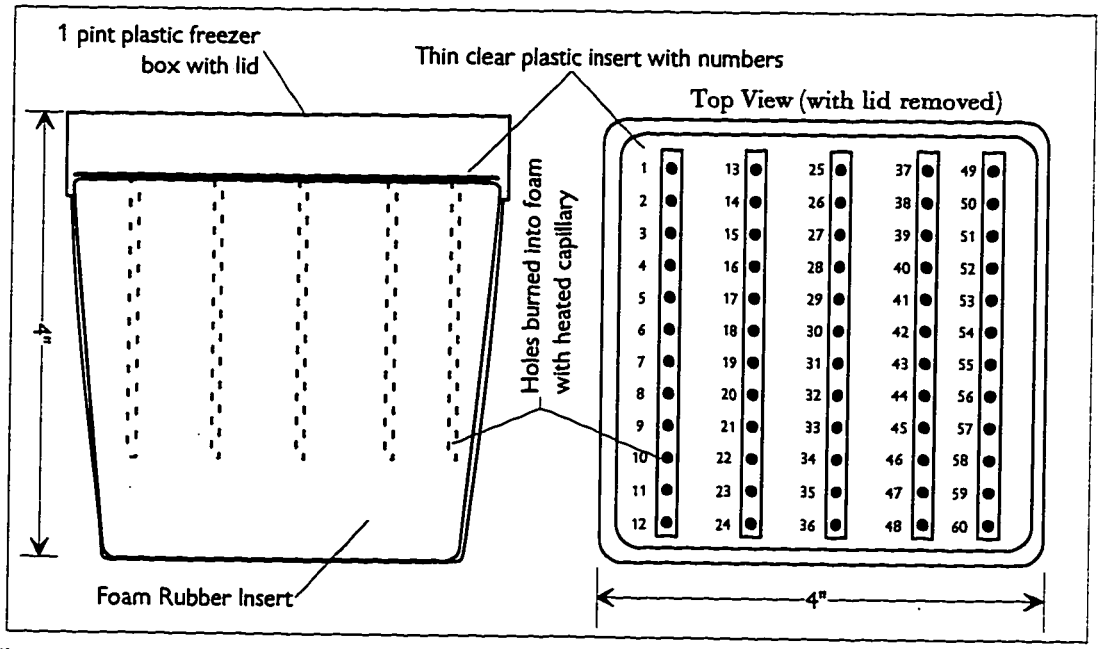


Figure 5.6 - Blood sample holder box. Contructed from a 1 pint white plastic freezer box, with a 3.5' foam insert. A clear plastic insert with numbers, and slots was placed on the foam rubber. A heated capillary burned holes in the foam, making slots for the blood samples.

capillary tube (Figure 5.6). A clear plastic sheet with numbers on it was used to keep track of the capillary tube numbers. Note that the insert and the holes in the foam were made in such a way that the plastic insert could only match the holes one way. Capillary tubes containing blood were sealed with Critoseal, and placed in the numbered holes in the storage box. The storage box was maintained in a small ice chest (with ice) until the samples could be centrifuged.

Ascelichthys rhodorus, Astyanax mexicanus* and *Enophrys bison: These fish were bled from the either the efferent or afferent branchial arteries as they pass over the gill arches (Figure 5.7). The fish were held in a nylon mesh (1/4" mesh veil material) to prevent the animal from escaping. 6" square pieces of the mesh were wrapped around the body and mouth of the fish, leaving one of the opercular openings exposed. For *Ascelichthys* and *Astyanax*, one person held the fish, and opened the opercular cover with forceps, but for the larger *Enophyrs*, this could be done by a single person,. The gill chamber was cleaned of mucus and excess water with a cotton-tipped applicator (swab). The branchial artery was punctured with a 27 gauge needle, and the blood was sucked up into a heparinized capillary tube. Gentle mouth suction was applied to a rubber hose attached to the capillary to draw in the small quantities of blood (Figure 5.5).

Lepidopsetta bilineata: These fish remain fairly quiescent when removed from the water. These fish were simply placed on the bleeding board, and the

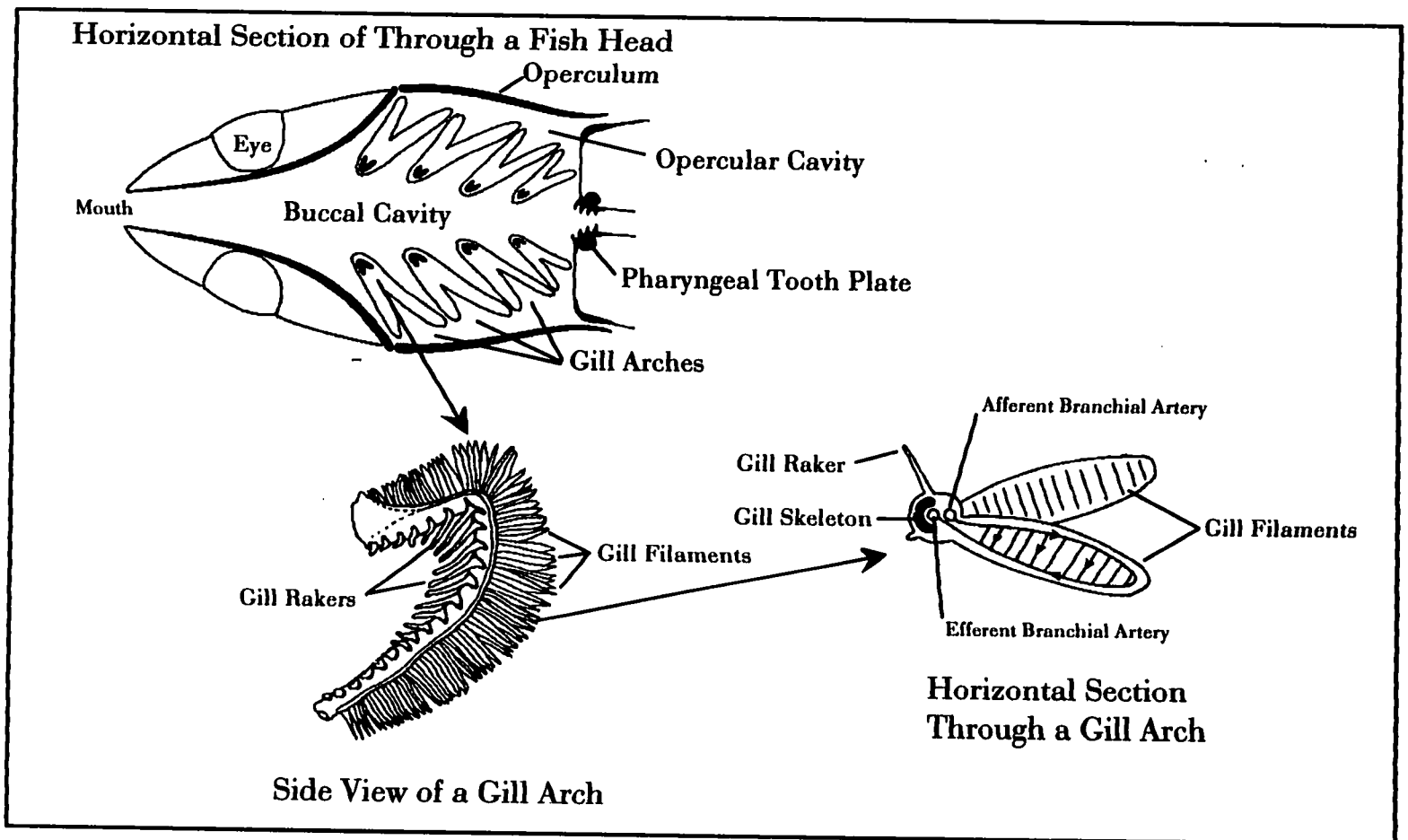


Figure 5.7 - Diagram of the positioning of gills within the head of a fish and details of the location of blood vessels used to take blood samples. Gills are oriented in a more or less vertical fashion within the buccal cavity. To take blood samples from the fish, the operculum is opened using a pair of forceps, and the Afferent or Efferent branchial arteries are punctured, and trhe blood is collected as it pools on the gill filaments.

opercular cover was lifted with a pair of forceps, and then the sample was taken as above for *Enophrys.*

Xiphister atropurpureus.: These pricklebacks are eel-like and extremely difficult to hold on to, so a holder was designed to make bleeding possible (Figure 5.8). The fish were coaxed into nylon mesh bags for storage, and when it was time to bleed the fish, the front few inches of the fish was allowed to escape the hole in the draw-string of the mesh bag, then the fish was clamped in the restraint device and the elastic straps were locked in place. The tension pressed the foam together with sufficient force to usually hold the fish in place, but not enough force to injure the fish. Once in the restraint device, the operculum could be lifted with forceps, and blood samples could be taken from the gills as above.

Capture Stress:

Fish were maintained in water filled white plastic buckets during the capture stress. The buckets were periodically agitated to prevent the fish from becoming acclimated to the bucket. *Xiphister* and *Ascellichthys* were maintained in nylon mesh bags within the buckets, and *Astyanax* was maintained in plastic bags in the buckets during the capture stress.

Radioimmunoassay of Cortisol

Tritiated-Cortisol Assay (used for *Ascelichthys, Astyanax, Lepidopsetta, Pleuronichthys,* and *Xiphister*): Plasma samples from 5-50 µl were diluted to 400 µl

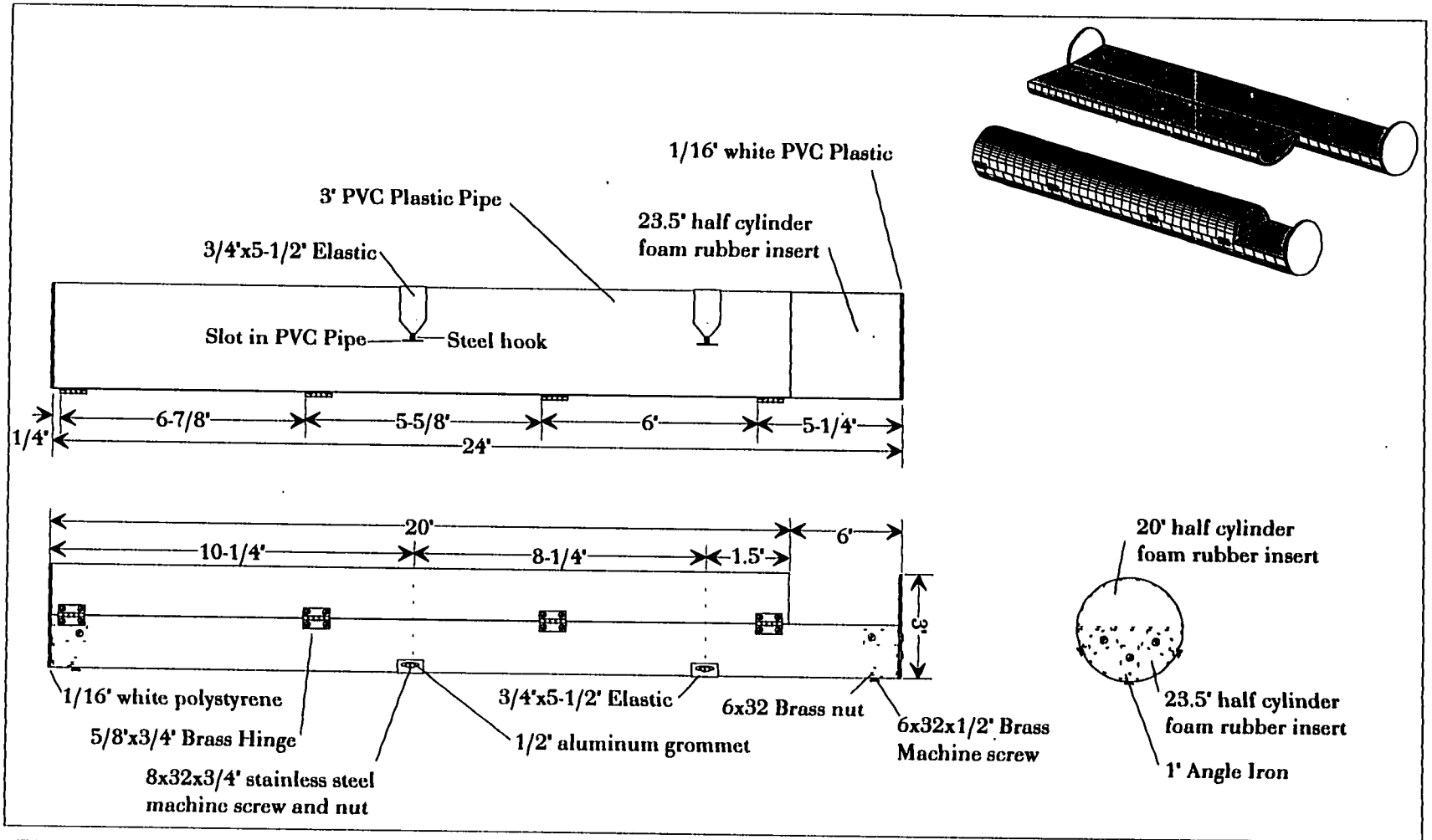


Figure 5.8 - Restraint device for bleeding pricklebacks. The device was contructed from a section of 3' PVC plastic pipe cut in half, and hinged. Two circular end pieces were affixed with glue and reinforced by angle irons to limit the possible motions of the fish. Two 1/2 cylindrical sections of firm foam rubber were fashioned to fit in the pipe, and 2 elastic straps added to apply pressure to the fish when in the holder.

with PBSG in a 15 ml conical centrifuge tube and 20 µl of labeled corticosterone (~2000 cpm - New England Nuclear) solution was added (to determine recoveries) and the samples were incubated overnight at 4°C. The samples were extracted overnight at 4°C with 5 ml of redistilled dichloromethane. The extracted steroid was transferred in the dichloromethane to test tube, and the dichloromethane was evaporated using a stream of nitrogen while the tubes incubated in a water-bath at 40°C. The steroid was allowed to resuspend in 550 µl of PBSG overnight at 4°C.

The following day, 100 µl from each sample was transferred to a scintillation vial to determine recoveries and 4.5 ml of scintillation fluid was added. 200 µl of each sample was transferred to each of 2 15x100mm disposable test tubes. Each tube was incubated with 100 µl of anti-cortisol antibody (Cat # 440A - diluted 1:50, Endocrine Sciences, Tarzana, CA), and 100 µl of tritiated cortisol solution (~10,000 cpm), and incubated overnight at 4°C. Separation of bound from free steroid was achieved by the addition of 0.5 ml of dextran-coated charcoal for 12 minutes at 4°C. All samples were then promptly centrifuged at 2,000 rpm for 10 minutes at 4°C in a Beckmann TJ-6 refrigerated centrifuge. Supernatants (containing bound cpm) were decanted into scintillation vials, 4.5 ml of scintillation cocktail added, and equilibrated overnight before counting in a Beckmann LS3801 system (for 10 min or 2% accuracy).

With each assay, 2 solvent blanks and a standard sample (containing 250 pg of cortisol) were taken through the entire assay procedure as a check on reliability criteria.

Data for the assays was collected from the scintillation counter by an IBM computer and plasma concentrations of cortisol were calculated using the program in Appendix B.

Iodinated-Cortisol Assay (used for *Enophrys*): This assay was done using a Clinical Assays™ GammaCoat™ Cortisol 125I RIA Kit (CA-1549, INCSTAR Corporation, Stillwater, MN) modified to increase sensitivity as follows:

1) The standard curve was diluted to the range of 2000 pg-7.8pg.
2) The volume used was increased to 25 µl, rather than 10 µl.
3) The volume of tracer solution added to each tube was decrease by 50% to 0.5 ml. This made the RIA curve more sensitive at the lower end.
4) Samples were briefly centrifuged after mixing to remove tracer/sample stuck to the walls of the tubes. This helped make the replicates more similar.
5) Incubation was done at 4°C overnight, rather than at 37°C for 45 minutes. This made a large increase in overall maximum binding (BMAX).
6) After incubation, the unbound tracer was removed using a vacuum aspirator and a drawn-out Pasteur pipette. The tubes were then briefly centrifuged to remove unbound tracer stuck to the walls of the tube, and the traces removed

with the vacuum aspirator. This removed most of the variability between replicates.

Statistics

All statistical analysis of the data was done using the computer program SYSTAT (Wilkinson, 1990). TPF responses over time for *Astyanax*, *Ascelichthys* and *Xiphister* were analyzed using one way ANOVAs. Responses to TPFs over time for Pleuronichthys, Lepidopsetta, and Enophrys were examined using one-way repeated measures ANOVAs. Post hoc comparisons between time points were made using routines for this purpose built into SYSTAT.

Results

Intertidal Fishes:

The intertidal sculpin *Ascelichthys rhodorus* showed a significant increase in cortisol in response to capture within 30 minutes (Figure 5.9: $F=19.955$, $df=1$, $P<0.001$) The peak level was reached around 50 minutes after capture, and began declining there after and was back to near basal by 140 minutes, even though the fish were periodically agitated within the bucket. Maximum cortisol levels were around 20 ng/ml, and basal cortisol levels were about 1 ng/ml.

The intertidal stichaeid *Xiphister atropurpureus* showed a similar trend to *A. rhodorus*, but there was no statistically significant effect of time after capture and

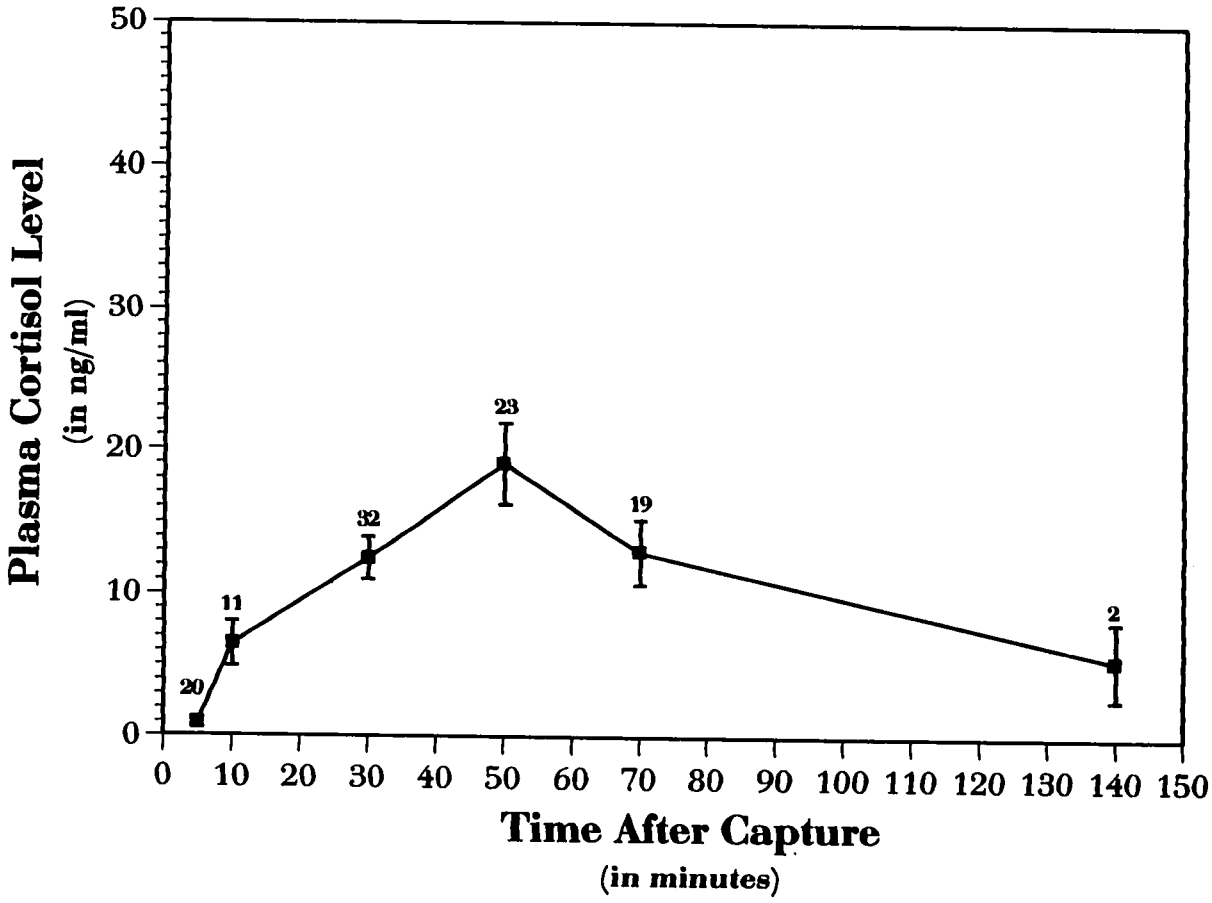


Figure 5.9 - Effect of capture and confinement on plasma cortisol in the intertidal cottid *Ascelichthys rhodorus*. Error bars represent +/- the standard error of the mean

cortisol levels (Figure 5.10: F=1.806, df=4, P=0.180). Maximum levels of cortisol were around 25 ng/ml.

Subtidal Fishes:

Enophrys bison did not show a glucocorticoid response to capture (Figure 5.11: F=0.788, df=2, P=0.487). In fact, there appears to be a trend toward decreasing cortisol levels, rather than increasing. The basal cortisol level was quite high, at around 80 ng/ml.

The pleuronectids *Lepidopsetta bilineata* and *Pleuronichthys coenosus* showed significant increases in plasma cortisol in response to capture (Figure 5.12: F=26.027, df=1, P=0.007 and F=72.254, df=1, P=0.014, respectively). Both had low (<10 ng/ml) basal cortisol concentrations.

Subterranean Fish:

Astyanax mexicanus showed a significant increase in cortisol levels in response to capture (Figure 5.13: F=24.196, df=2, P=0.001). Basal cortisol levels were under 10 ng/ml, and maximum level within the first 35 minutes of capture was >100 ng/ml. This much more similar to the pleuronectids than to that of the intertidal fishes.

Discussion

Both species of subtidal pleuronectids (*Pleuronichthys coenosus* and

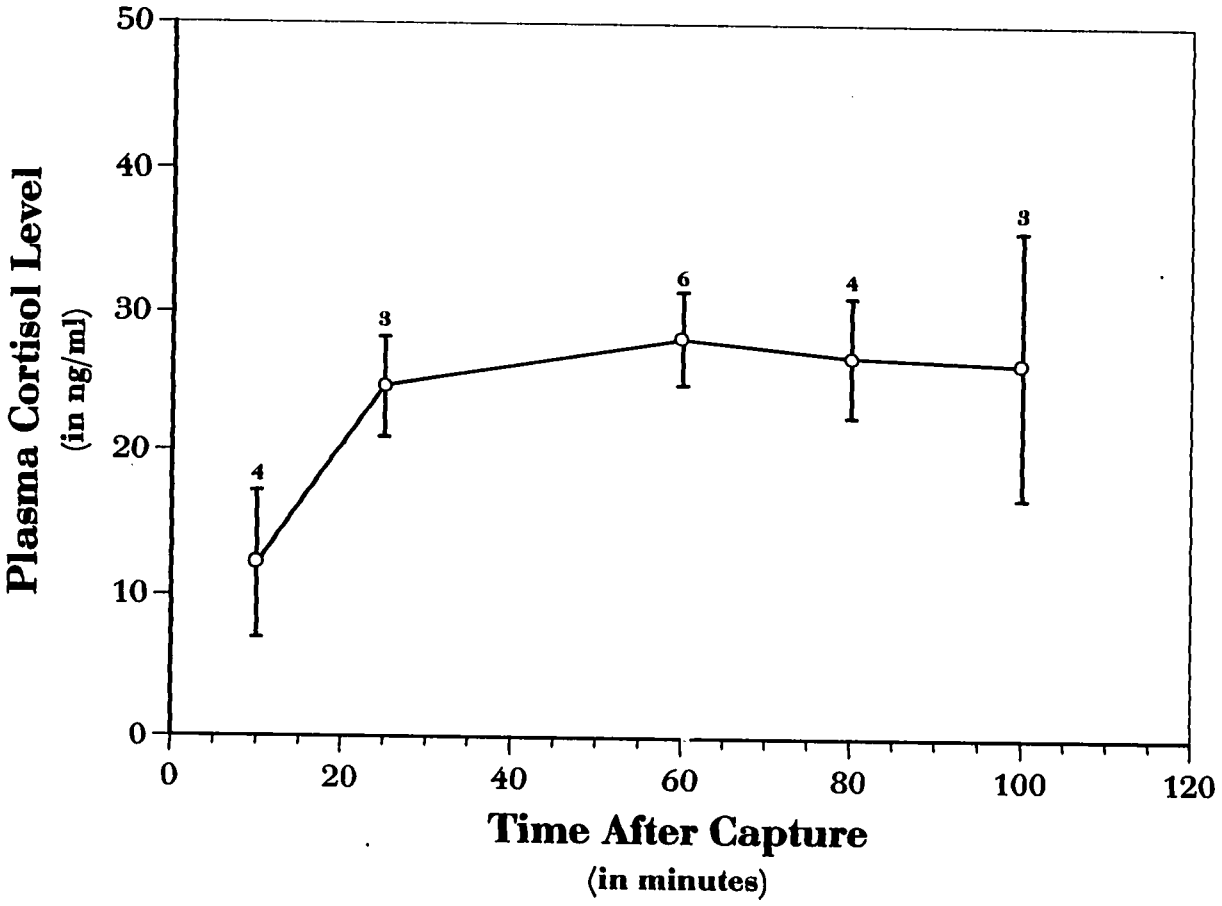


Figure 5.10 - Effect of capture and confinement on plasma cortisol in the intertidal stichaei d*Xiphister atropurpureus*. Error bars represent +/- the standard error of the mean

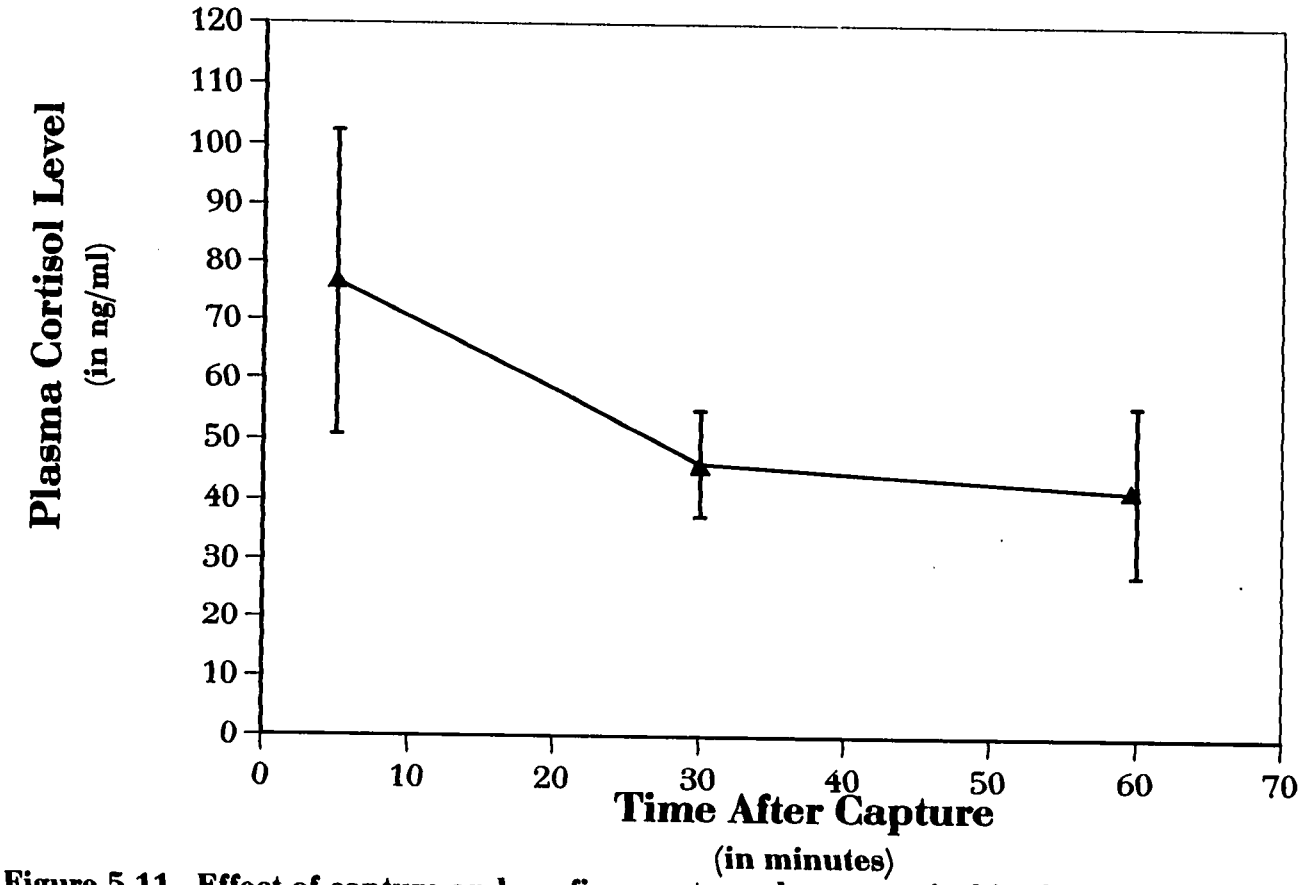


Figure 5.11 - Effect of capture and confinement on plasma cortisol in the subtidal cottid *Enophrys bison*. Each point represents 5 fish. Each fish was bled at each time point. Error bars represent +/- the standard error of the mean

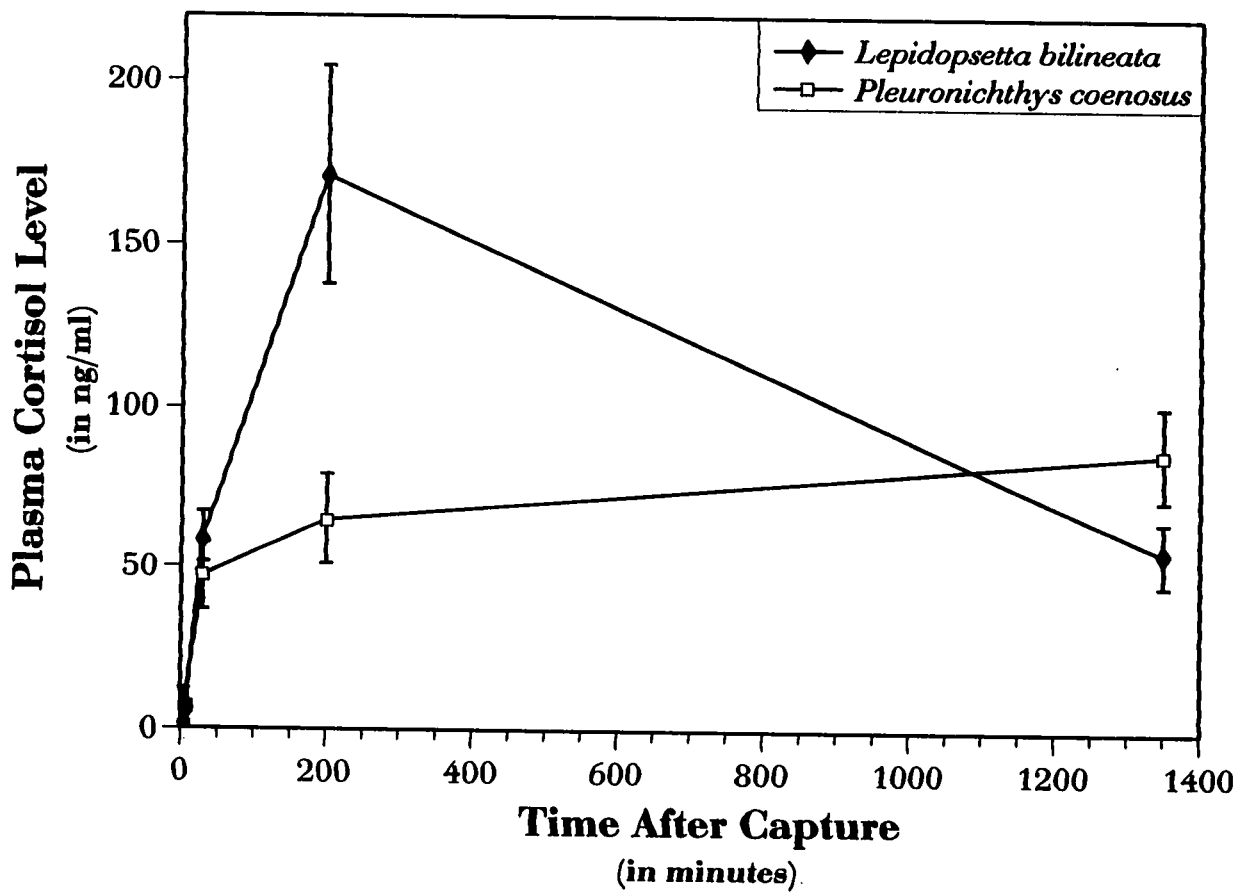


Figure 5.12 - Effect of capture and confinement on plasma cortisol in 2 subtidal pleuronectids. Each fish was bled at every time point. For *Pleuronichthys coenosus*, n=3 and for *Lepidopsetta bilineata*, n=5. Error bars represent +/- the standard error of the mean

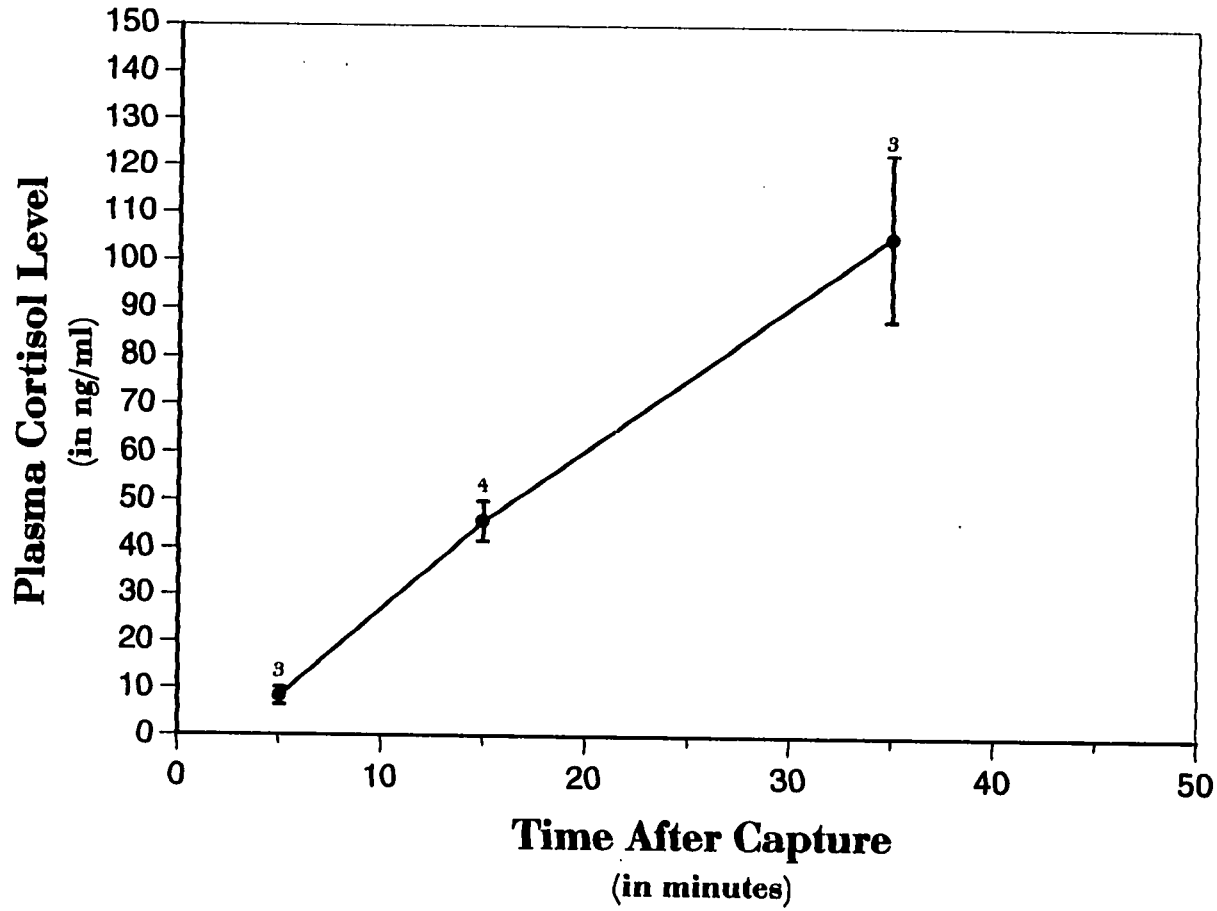


Figure 5.13 - Effect of capture and confinement on plasma cortisol in a cave living characid *Astyanax mexicanus*. Error bars represent +/- the standard error of the mean

Lepidopsetta bilineata) showed well developed adrenocortical responses in response to capture/confinement as have been seen in other pleuronectids (Wingfield, 1973). Basal cortisol levels for both pleuronectids was low as is normal in most animals. Unfortunately, there are apparently no intertidal pleuronectids, so it was not possible to compare this response to an intertidal form in the same family.

Both intertidal fishes examined (*Xiphister atropurpureus*: Stichaeidae and *Ascellichthys rhodorus*: Cottidae) showed significant responses to capture/confinement, but these levels were less than 50% of the maximal levels obtained in the pleuronectids (see figure 5.14). The response of *Ascellichthys* to the TPF decayed very rapidly, reaching its maximum at approximately 50 minutes, and reaching basal by 140 minutes after capture. In other fishes, elevated cortisol levels persist for hours after capture (Pickering and Pottinger, 1989), and if maintained in over-crowded or otherwise poor conditions, elevated cortisol levels can persist for at least 25 days (Pickering and Stewart, 1984) and possibly up to 6 months (Tam *et al.*, 1987). It is possible that since *Ascellichthys* normally is found in small pools under rocks in the intertidal zone that confinement in nylon bags within a bucket of water is not perceived as a TPF. Attempts were made to disturb the fish within the bucket by periodically hitting the bucket, and stirring the fish about, but this may not have been enough to maintain stimulation of the adrenocortical response. It should be noted that in the *Ascellichthys* that were serially bled (and thus handled far more), there was also only an attenuated

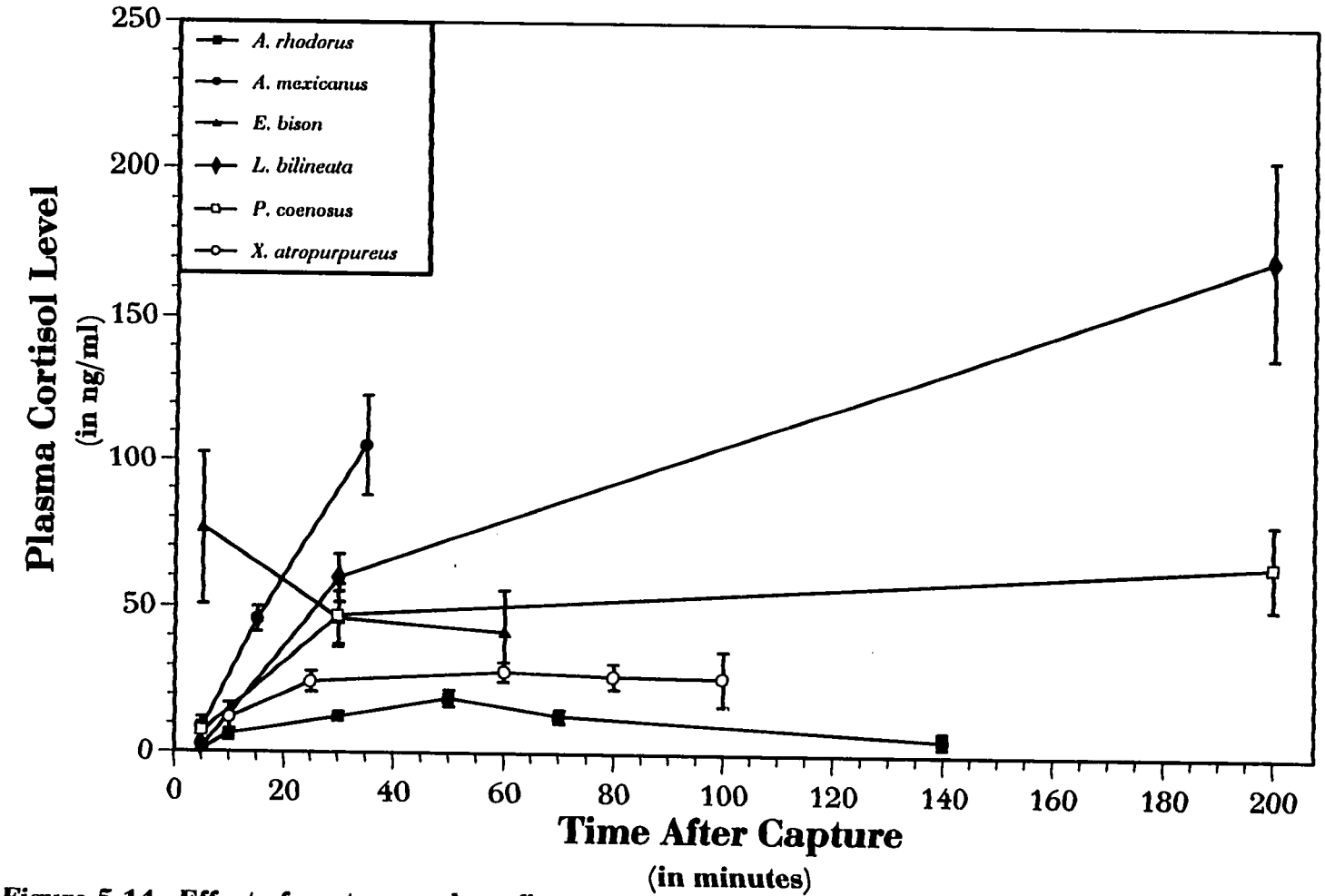


Figure 5.14 - Effect of capture and *confinement* on plasma cortisol in several fishes Error bars represent +/- the standard error of the mean

response to capture/confinement (personal observation), so additional handling does not increase the magnitude or duration of the response.

The adrenocortical response the of the intertidal prickleback *Xiphister atropurpureus*, while significant, was of a very low amplitude (approximately a 2 fold increase over the <10 minute sample). A similar attenuation of the response to TPFs is seen during the breeding in birds breeding in the Arctic (Wingfield *et al.*, 1994a, 1994b), and also birds breeding in the Sonoran Desert (Wingfield *et al.*, 1992). While the attenuation of the adrenocortical responses in these birds occurred while breeding, the attenuation in *Xiphister* was not due to breeding constraints, as samples were not collected from breeding fish (*Xiphister* breeds in the early spring, and males guard and "fan" a ball of eggs in the intertidal zone).

Surprisingly, the subtidal sculpin *Enophrys bison* did not show a response to capture/confinement. They also had remarkably high basal cortisol levels which were larger than the maximum levels for both intertidal fishes, and for *Pleuronichthys*. This elevated basal cortisol level may be in response to environmental pollutants, as the fish were collected near an abandon commercial oil dock. While possible, this is not likely, as pleuronectids caught in the same area had low basal cortisol levels. Also the pilings of the oil dock have a health and diverse community of both invertebrates and fishes (personal observation). The lack of adrenocortical response to capture/confinement has been indicated by preliminary data from two other subtidal sculpins (*Myxocephalus*

polyacanthocephalus and *Hemilepidotus hemilepidotus*, unpublished data). While the lack of response of the subtidal sculpins was not expected, there may be an evolutionary explanation for this lack of response. Both the subtidal and intertidal sculpins show a number of adaptations useful for intertidal life (such as reduced numbers of scales to increase transdermal respiration) so it is possible that this group evolved in the unpredictable intertidal zone where the adrenocortical response became attenuated, and later expanded out into the subtidal zone, and never regained the responsiveness to TPFs. It is also possible that these fish are not secreting cortisol in response to TPFs. Many other vertebrates secrete corticosterone, and it may be possible that corticosterone or some other adrenal steroid is being released. Further investigations into the steroid biochemistry of Cottids need to be done to answer this question.

The cave fish *Astyanax mexicanus* show the most rapid increase in cortisol of any of the fishes examined. Part of this increase was undoubtedly due to the higher temperature of the fish (20°C for *Astyanax* vs. approximately 10°C for the marine fishes.) It may also be due to sensitization of the cave fishes to TPFs, as the caves they are adapted to may be very predictable with limited numbers of TPFs. There is ample evidence that exposure to noxious stimuli can lead to reduced overall response to TPFs, even to TPFs different than those they have become acclimated to (*e.g.*, Camp and Robinson, 1988).

Both the intertidal fishes showed the expected limited adrenocortical responses for animals living in a relatively unpredictable environment (see chapter 2). Unfortunately, it was not possible to obtain samples from either a subtidal Stichaeid or an intertidal pleuronectid. The subtidal cottid did not respond as expected, but this may be due to this family having evolved in the intertidal zone.

The Cave fish Astyanax did not show the expected attenuation of the adrenocortical response for a fish evolving in a very predictable environment. As these fish have only recently entered caves and have a epigean form in the same species, they may not have adapted to cave conditions enough to show an adrenocortical attenuation. Further investigations into more troglodytic fishes such as *Ambylopsis*, as well as getting samples from the epigean form of *Astyanax mexicanus*. It would also be interesting to investigate the responses of some of the troglodytic salamanders found in caves of the south eastern United States.

Bibliography

1. Alexandrova, M. and Farkas, P. (1992). Stress-induced changes of glucocorticoid receptor in rat liver. *J. Steroid. Biochem. Mol. Biol.* **42**(5):493-498.

2. Ambler, M.P., and Chapman, V.J. (1950). A quantitative study of some factors affecting tide pools. *Trans. Roy. Soc. New Zealand* **78**(4):394-409.

3. Anderson, D.J. (1992). Masked Booby. In **The Birds of North America** (A. Poole, P. Stettenheim and F. Gill, Eds.), Vol. 2, No. 73, 16 pp. Academy of Natural Sciences, Philadelphia.

4. Astheimer, L.B., Buttemer, W.A., and Wingfield, J.C. (1992). Interactions of corticosterone with feeding, activity, and metabolism in passerine birds. *Ornis Scand.* **23**:355-365.

5. Axelrod, J. and Reisine, T.D. (1884). Stress hormones: Their interaction and regulation. *Science* **224**:452-459.

6. Babcock, R.C., Mundy, C.N., and Whitehead, D. (1994). Sperm diffusion models and *in situ* confirmation of long-distance fertilization in the free-spawning asteroid *Acanthaster planci*. *Biol. Bull.* **186**(1):17-28.

7. Ball, G.F., and Wingfield, J.C. (1987). Changes in plasma levels of sex steroids in relation to multiple broodedness and nest site density in male starlings. *Physiol. Zool.* **60**:191-199.

8. Bateman, A., Singh, A., Kral, T., and Solomon, S. (1989). The immune-hypothalmic-pituitary-adrenal axis. *Endocrin. Rev.* **10**:92-112.

9. Best, C.H., and Solandt, D.Y. (1940). Studies in experimental shock. *Canad. Med. Assoc. J.* **43**:206 (1940).

10. Blanchard, B.D. (1941). The white-crowned sparrows (*Zonotrichia leucophrys*) of the Pacific seaboard: environment and annual cycle. *Univ. Calif. Publ. Zool.* **46**:1-178.

11. Briskie, J.V. (1992). Smith's Longspur. In **The Birds of North America** (A. Poole, P. Stettenheim and F. Gill, Eds.), Vol. 1, No. 34, 16 pp. Academy of Natural Sciences, Philadelphia.

12. Buttemer, W.A. (1985). Energy relations of winter roost-site utilization by American goldfinches (*Carduelis tristis*). *Oecologia* **68**(1):126-132

13. Cabe, P.R. (1992). European Starling. In **The Birds of North America** (A. Poole, P. Stettenheim and F. Gill, Eds.), Vol. 2, No. 48, 24 pp. Academy of Natural Sciences, Philadelphia.

14. Cain, J.R., and Lien, R.J. (1985). A model for drought inhibition of bobwhite quail (Colinus virginianus) reproductive systems. *Comp. Biochem. Physiol.* **82A**:925-930.

15. Camp, D.M., and Robinson, T.E. (1988). Susceptibility to sensitization. II. The influence of gonadal hormones on enduring changes in brain monoamines and behavior produced by the repeated administration of D-amphetamine or restraint stress. *Behav. Brain Res.* **30(1)**:69-88.

16. Carey, L.C., Cloutier, C.T., and Lowery, B.D. (1971). Growth hormone in adrenocortical response to shock and trauma in the human. *Ann. Surg.* **174**:451-460.

17. Carsia, R.V. and Weber, H. (1986). Genetic-dependent alterations in adrenal stress response and adrenocortical cell function of the domestic fowl (*Gallus domesticus*). *Proc. Soc. Exp. Biol. Med.* **183(1)**:99-105.

18. Chapin, J.P. (1954). The calendar of wide awake fair. *Auk* **71**: 1-15.

19. Chapman, P. (1993). **Caves and Cavelife**. Harper Collins Publishers, London.

20. Cherel, Y., Leloup, J., and Le Maho, Y. (1988). Fasting in king penguin: II. Hormonal and metabolic changes during molt. *Amer. J. Physiol.* **254(2/2)**:R178-R184.

21. Cherel, Y. and Le Maho, Y. (1985). Five months of fasting in King Penguin chicks: body mass loss and fuel metabolism. *Amer. J. Physiol.* **249**:R387-R392.

22. Cherel, Y., Robin, J.-P., Walch, O., Karmann, H., Netchitailo, P., and Le Maho, Y. (1988). Fasting in king penguin: I. Hormonal and metabolic changes during breeding. *Amer. J. Physiol.* **254(2/2)**:R170-R177.

23. Cherel, Y., Stahl, J.-C., and Le Maho, Y. (1987). Ecology and physiology of fasting in King Penguin chicks. *Auk* **104**:254-262.

24. Collu, R., Gibb, W., and Ducharme, J.R. (1984). Effects of stress on the gonadal function. *J. Endo. Invest.* 7:529-537.

25. Connell, J.H. (1978). Diversity in tropical rain forests and coral reefs. *Science* **199**:1302-1310.

26. Conover, D.O. and Fleisher, M.H. (1987). Temperature-sensitive period of sex determination in Atlantic silverside, *Menidia menidia. Can. J. Fish. Aquat. Sci.* **43(3)**:514-520.

27. Crews, D., Bergeron, J.M., Bull, J.J., Flores, D., Tousignant, A., Skipper, J.K., and Wibbels, T. (1994). Temperature-dependent sex determination in reptiles: proximate mechanisms, ultimate outcomes, and practical applications. *Devel. Genetics* **15(3)**:297-312.

28. Daniel, M.J., and Boyden, C.R. (1975). Diurnal variation in physico-chemical conditions within intertidal rockpools. *Field Studies* **4**:161-176.

29. Dardevet, D., Sornet, C., Taillandier, D., Savary, I., Attaix, D., and Grizard, J. (1995). Sensitivity and protein turnover response to glucocorticoids are different in skeletal muscle from adult and old rats. Lack of regulation of the ubiquitin-proteasome proteolytic pathway in aging. *J. Clin. Invest.* **96**(5):2113-2119.

30. Dawkins, R. (1982). **The Extended Phenotype. The Gene as the Unit of Selection.** W.H. Freeman and Co., Oxford.

31. De-Kloet, E.R., Rosenfeld, P. Van Eekelen, J.A., Sutanto,W., and Levine, S. (1988). Stress, glucocorticoids and development. *Prog. Brain. Res.* **73**:101-120.

32. Dohms, J.,E. and Metz, A. (1991). Stress–mechanisms of immuno-suppression. *Vet. Immunol. Immunopathol.* **30(1)**:89-109.

33. Dufty, A.M.Jr., and Wingfield, J.C. (1986a). Temporal patterns of circulating LH and steroid hormones in a brood parasite, the brown headed cowbird, *Molothrus ater.* I. Males. *J. Zool. (London)* **208**:191-203.

34. Dufty, A.M.Jr., and Wingfield, J.C. (1986b). Temporal patterns of circulating LH and steroid hormones in a brood parasite, the brown headed cowbird, *Molothrus ater.* II. Females. *J. Zool. (London)* **208**:205-214.

35. Dunlop. D. (1963). Eighty-six cases of Addison's disease. *Brit. Med. J.* **2**:887.

36. Eckert, R., Randall, D., and Augustine, G. (1988). **Animal Physiology.** W.H. Freeman and Co., New York.

37. Edgar, B.A., Kiehle, C.P., and Schubiger, G. (1986). Cell cycle control by nuclear cytoplasmic ratio in early *Drosophila* development. *Cell* **44**(2):365-372.

38. Eichele, G. (1989). Retinoids and vertebrate limb pattern formation. *Trends in Genetics* **5**(8):246-251

39. Ferguson, M.W.J., and Joanen, T. (1983). Temperature-dependent sex determination in *Alligator mississippiensis. J. Zool.* **200**(2):143-177.

40. Finlay, W.E.I, and McKee, J.I. (1982). Serum cortisol in severely stressed patients. *Lancet* **1**:1414-1415.

41. Flaherty, C.F. and Rowan, G.A. (1989). Rats (*Rattus norvegicus*) selectively bred to differ in avoidance behavior also differ in response to novelty stress, in glycemic conditioning, and in reward contrast. *Behav. Neural. Biol.* **51**(2):145-164.

42. Freeman, B.M., and Flack, I.H. (1980). Effects of handling on plasma corticosterone concentrations in immature domestic fowl. *Comp Biochem. Physiol.* **66A**:77-81.

43. Freeman, B.M., Manning, C.C., and Flack, I.H. (1984). Changes in plasma corticosterone concentrations in the water-deprived fowl, *Gallus domesticus. Comp. Biochem. Physiol.* **79A**:457-458.

44. Freund, R.K., Martin, B.J., Jungschaffer, D.A., Ullman, E.A., Collins, A.C. (1988). Genetic differences in plasma corticosterone levels in response to nicotine injection. *Pharmacol. Biochem. Behav.* **30**(4):1059-1064.

45. Fry, F.E., Brett, J.R., and Clawson (1942). Lethal limits of temperature for young goldfish. *Rev. Canad. Biol.* **1**:50-56.

46. Ganning, B. (1971). Studies on chemical, physical and biological conditions in Swedish Rockpool ecosystems. *Ophelia* **9**:51-105.

47. Gibson, R.N. (1993). Intertidal teleosts: life in a fluctuating environment. In **Behavior of Teleost Fishes, 2nd Edition** (T.J. Pitcher, ed.) Chapman and Hall, New York. pp 513-536.

48. Gilbert, J.J., (1980). Developmental polymorphism in the rotifer *Asplanchna sieboldi. American Scientist* **68**(6):636-646.

49. Gilbert, S.F. (1991). **Developmental Biology. 3rd Edition.** Sinauer Associates, Sunderland, Mass.

50. Gill, A. (1962). **Introduction to the Theory of Finite-State Machines.** McGraw-Hill Book Company, Inc., New York, New York.

51. Gratto-Trevor, C.L. (1992). Semipalmated Sandpiper. In **The Birds of North America** (A. Poole, P. Stettenheim and F. Gill, Eds.), Vol. 1, No. 6, 20 pp. Academy of Natural Sciences, Philadelphia.

52. Greenburg, N., and Wingfield, J.C. (1987). Stress and reproduction: reciprocal relationships. In **Reproductive Endocrinology of Fishes, Amphibians and Reptiles.** D.O. Noriss and R.E. Jones eds.), pp 389-426, Wiley, New York.

53. Gross, W.B. and Siegel, P.B. (1985). Selective breeding of chickens for corticosterone response to social stress. *Poult. Sci.* **64(12)**:2230-2233.

54. Guderley, H. (1990). Functional significance of metabolic responses to thermal acclimation in fish muscle. *Amer. J. Physiol.* **259(2-2)**:R245-R252.

55. Hadley, M.E. (1996). **Endocrinology,** 4th edition. pp 314-336. Prentice Hall, Inc., Upper Saddle River, NJ.

56. Hahn, T.P. (1993). Integration of environmental cues to time reproduction in an opportunistic breeder, the red crossbill (*Loxia curvirostra*). Ph.D. Thesis, University of Washington.

57. Hall, B.K., and Kalliecharan, R. (1975). The effects of exogenous cortisone acetate on development (especially skeletal development) and on circulating levels of corticosteroids in chick embryos. *Teratology* **12**:111-120.

58. Hairston, N.G. Jr. and Olds, E.J. (1986). Partial photoperiodic control of diapause in three populations of the freshwater copepod *Diaptomus sanguineus. Biol. Bull.* **171**:135-142.

59. Harvey, S., Phillips, J.G., Rees, A., and Hal, T.R. (1984). Stress and adrenal function. *J. Exp. Zool.* **232**:633-646.

60. Harvell, C.D. (1992). Inducible defenses and allocation shifts in a marine byrozoan. *Ecology* **73(5)**:1567-1576.

61. Hastings, M.H., Vance, G., and Maywood, E. (1989). Some reflections on the phylogeny and function of the pineal. *Experientia* **45(10)**:903-909.

62. Hochachka, P.W., and Somero, G.N. (1973). **Strategies of Biochemical Adaptation**. W.B. Saunders Co., Phildelphia, PA.

63. Holder, K., and Montgomerie, R. (1992). Rock Ptarmigan. **In The Birds of North America** (A. Poole, P. Stettenheim and F. Gill, Eds.), Vol. 2, No. 51, 24 pp. Academy of Natural Sciences, Philadelphia.

64. Holmes, W.N., and Phillips, J.G. (1976). The adrenal cortex in birds. **In: General, Comparative and Clincal Endocrinology of the Adrenal Cortex. Vol. 1.** Chester-Jones, and I.W. Henderson, eds. Academic Press, NewYork, pp 293-420.

65. Horn, M.H., and Gibson, R. (1988). Intertidal Fishes. *Sci. Amer.* **258(1)**:64-70.

66. Houston, A.I., and McNamara, J.M. (1992). Phenotypic plasticity as a state-dependent life-history decision. *Evol. Ecol.* 6:243-253.

67. Idelman, S., and Marie, C. (1991). Ontogenesis of the pituitary adrenal axis in chick embryo. In **Current Trends in Comparative Endocrinology. Volume 1** (Lofts, B., and Holmes, W.W.,, eds). Hong Kong University Press. pp 455-456.

68. Ingles, L.G. (1965). **Mammals of the Pacific States. California, Oregon, and Washington**. Stanford University Press, Stanford, CA.

69. Irwin, M. (1994). Stress-induced immune suppression: role of brain corticotropin releasing hormone and autonomic nervous system mechanisms. *Adv. Neuroimmunol.* **4(1)**:29-47.

70. Jacobson, A.G., and Sater, A.K. (1988). Features of embryonic induction. *Development* **104(3)**:341-359.

71. Janzen, F.J. and Paukstis, G.L. (1991). Environmental sex determination in reptiles: ecology, evolution, and experimental design. *Quart. Rev. Biol.* **66(2)**:149-179.

72. Johnston, R.F. (1992). Rock Dove. **In The Birds of North America** (A. Poole, P. Stettenheim and F. Gill, Eds.), Vol. 1, No. 13, 16 pp. Academy of Natural Sciences, Philadelphia.

73. Judd, J.R. (1917). Observations on gas-bacillus infection in France. *Surg., Gynec. and Obst.* **25**:113.

74. Kalliecharan, R., and Hall, B.K. (1974). A developmental study of the levels of progesterone, corticosterone, cortisol, and cortisone circulating in plasma of chick embryos. *Gen. Comp. Endo.* **24**:364-372.

75. Kalliecharan, R., and Hall, B.K. (1976). A developmental study of progesterone, corticosterone, cortisol, and cortisone in the adrenal glands of the embryonic chick. *Gen. Comp. Endo.* **30**:404-409.

76. Kaplan, R.H., and Cooper, W.S. (1984). The evolution of developmental plasticity in reproductive characteristics: An application of the "adaptive coin-flipping" principle. *Am. Nat.* **123(3)**: 393-410

77. Krulich , L., Hefco, E., Illner, P, and Read, C.B. (1974). The effects of acute stress on the secretion of LH, FSH, prolactin and GH in the normal rat, with comments on their statistical evaluation. *Endocrinology* **96**:85-92.

78. Le Ninan, F., Cherel, Y., Sardet, C., and Le Maho, Y. (1988). Plasma hormone levels in relation to lipid and protein metabolism during prolonged fasting in King Penguin chicks. *Gen. Comp. Endo.* **71**:331-337.

79. Levins, R. (1968). **Evolution in Changing Environments. Some Theoretical Explorations.** Princeton University Press. Princeton, New Jersey.

80. Linkhart, T.A., Clegg, C.H., and Hauschka, S.D. (1980). Control of mouse myoblast commitment to terminal differentiation by mitogens. *J. Supramolecular Struct.* **14**:483-498.

81. Lockwood, S.F., Holland, B.S., Bickham, J.W., Hanks, B.G., and Bull, J.J. (1991). Intraspecific genome size variation in a turtle (*Trachemys scripta*) exhibiting temperature-dependent sex determintation. *Can. J. Zool.* **69(9)**:2306-2310.

82. Lofts, B., and Murton, R.K. (1968). Photoperiodic and physiological adaptations regulating avian breeding cycles and their ecological significance. *J. Zool. (London)* **155**:327-394.

83. Lowther, P.E., and Cink, C.L. (1992). House Sparrow. In **The Birds of North America** (A. Poole, P. Stettenheim and F. Gill, Eds.), Vol. 1, No. 12, 20 pp. Academy of Natural Sciences, Philadelphia.

84. MacArthur, R.H. (1964). Environmental factors affecting species diversity. *Am. Nat.* **98**:387-397.

85. MacArthur, R.H., and MacArthur, J.W. (1961). On bird species diversity. *Ecology* **42**:594-598.

86. MacArthur, R.H., and Preer, J. (1962). On bird species diversity II. Prediction of bird census from habitat measurements. *Am. Nat.* **96**:167-174.

87. Mangel, M., and CLark, C.W. (1986). Towards a unified foraging theory. *Ecology* **67(5)**:1127-1138.

88. Manuwal, D.A. and Thoresen, A.C. (1992). Cassin's Auklet. **In The Birds of North America** (A. Poole, P. Stettenheim and F. Gill, Eds.), Vol. 2, No. 50, 20 pp. Academy of Natural Sciences, Philadelphia.

89. Martin, T.E. (1988). Processes organizing open-nesting bird assemblages: competition or predation? *Evol. Ecol.* **2**:37-50.

90. Meaney, M.J., Bhatnagar, S., Diorio, J., Larocque, S., Francis, D., O'Donnell, D., Shanks, N., Sharma, S., Smythe, J., and Viau, V. (1993a). Molecular basis for the development of individual differences in the hypothalamic-pituitary-adrenal stress response. *Cell. Mol. Neurobiol.* **13(4)**:321-347.

91. Meaney, M.J., Bhatnagar, S., Larocque, S., McCormick, C., Shanks, N., Sharma, S., Smythe, J., Viau, V., and Plotsky, P. M. (1993b). Individual differences in the hypothalamic-pituitary-adrenal stress response and the hypothalamic CRF system. *Ann. N. Y. Acad. Sci.* **697**:70-85.

92. Meaney, M.J., Diorio, J., Francis, D., Widdowson, J., LaPlante, P., Caldji, C., Sharma, S., Reckl, J.R., and Plotsky, P.M. (1996). Early environmental regulation of forebrain glucocorticoid receptor gene expression: Implications for adrenocortical responses to stress. *Dev. Nerusosci.* **18**:49-72.

93. Mewaldt, L.R., and King, J.R. (1977). The annual cycle of white-crowned sparrows (*Zonotrichia leucophrys nuttallii*) in coastal California. *Condor* **79**: 445-455.

94. Miller, A.H. (1962). Bimodal occurrence of breeding in an equatorial sparrow. *Proc. Natl. Acad. Sci. U.S.A.* **48**:396-400.

95. Mills, F.J. (1985). The endocrinology of stress. *Aviation, Space, and Env. Med.* **56**:642-650.

96. Moon, V.H., Kornblum, K., and Morgan, D.R. (1940). Pathology of irradiation sickness: a new method of inducing shock. *Proc. Soc. Exper. Biol. and Med.* **43**:305

97. Morris, S., and Taylor, A.C. (1983). Diurnal and seasonal variation in physico-chemcal conditions within intertidal rock pools. *Est. Costal and Shelf Sci.* **17**:339-355,

98. Mueller, A.J. (1992). Inca Dove. In **The Birds of North America** (A. Poole, P. Stettenheim and F. Gill, Eds.), Vol. 1, No. 28, 12 pp. Academy of Natural Sciences, Philadelphia.

99. Munk, A., Guyre, P.M., and Holbrook, N.J. (1984). Physiological functions of glucocorticiods in stress and their relation to pharmacological actions. *Endocrine Reviews* **5**:25-44.

100. Mustafa, T.M., and Hodgeson, C.J. (1984). Observations on the effect of photoperiod on the control of polymorphism in *Psylla pyricola. Physiol. Entomol.* **9**:207-213.

101. Myers, R.F. (1991). **Micronesian Reef Fishes. A Practical Guide to the Identification of the Coral Reef Fishes of the tropical Central and Western Pacific.** Coral Graphics, Barrigada, Guam.

102. Newport, J., and Kirschner, M. (1982a). A major developmental transition in early *Xenopus* embryos: I. Characterization and timing of cellular changes at the midblastula stage. *Cell* **30**:675-686.

103. Newport, J., and Kirschner, M. (1982b). A major developmental transition in early *Xenopus* embryos: II. Control of the onset of transcription. *Cell* **30**:687-696.

104. Olitzki, L., Avinery, S., and Koch, P.K. (1942). The hypothermic and adrenohemorhagic effects of bacterial vaccines. *J. Immunol.* **45**:237.

105. O'Malley, B.W., and Tsai, M.J. (1992). Molecurlar pathways of steroid rescptor action. *Biol. Repro.* **46(2)**:163-167.

106. Pickering, A.D., and Pottinger, T.G. (1989). Stress responses and disease resistance in salmonid fish: Effects of chronic elevation of plasma cortisol. *Fish Physiol. Biochem.* **7**:253-258.

107. Pickering, A.D., Pottinger, T.G., Carragher, J., and Sumpter, J.P. (1987). The effects of acute and chronic stress on the levels of reproductive hormones in the plasma of mature male brown trout, *Salmo trutta* L. *Gen Comp. Endo.* **68**:249-259.

108. Pickering A.D. and Stewart, A. (1984). Acclimation of the interrental tissue of the brown trout, *Salmo trutta* L., to chronic stress. *J Fish Biol.* **24**:731-740.

109. Pielou, E.C. (1969). **An Introduction to Mathematical Ecology**. Wiley Interscience, New York, 286 pp.

110. Pielou, E.C. (1975). **Ecological Diversity.** Wiley Interscience, New York, 165 pp.

111. Pielou, E.C. (1994). **A Naturalists Guide to the Actic**. The University of Chicgo Press. Chicago, Illinois.

112. Poulson, T.L. (1964). Animals in aquatic environments: animals in caves. Chapter 47 in **Handbook of Physiology. Section IV. Adaptations to the Environment.** ed. Dill, D.B., Adloph, E.F., and Wilber, C.G. American Physiological Society. Washington, D.C.

113. Primmett, D.R.N., Norris, W.E., Carlson, G.J., Keynes, R.J., and Stern, C.D. (1989). Periodic segmental anomalies induced by heat shock in the chick embryo as associated with the cell cycle. *Development* **105**:119-130.

114. Prosser, C.L. (1986). **Adaptaionl Biology: Molecules to Organisms.** John Wiley & Sons. New York, New York.

115. Ramade, F., and Bayle, J.-D. (1980). Evaluation de l'activite hypophyso-corticosurrenalienne basale et de sa reponse au stress au cours de la periode orecedant et suivant l'eclosion chex le pigeon. *J. Physiol. Paris* **76**:283-287.

116. Ricklefs, R.E. (1987). Community diversity: relative roles of local and regional processes. *Science* **235**:167-171.

117. Robertson, D.R. (1975). Social control of sex reversal in a coral-reef fish. *Science* **177**:1007-1009.

118. Romanoff, A.L. (1941). Development of homeothermy in birds. *Science* **94(2435)**:218-219.

119. Romanoff, A.L. (1960). **The avian embryo; structural and functional development**. Macmillan, New York.

120. Romer, A.S. (1962). **The Vertebrate Body**. 3rd Edition. p. 358. W.B. Saunders Co. Philadephia.

121. Romero, L.M., Plotsky, P.M., and Sapolsky, R.M. (1993). Patterns of arenocoritotropin secretagog release with hypoglycemia, novelty, and restraint after colchicine blockade of axonal transport. *Endocrinology* **132**(1): 199-204.

122. Ross, R.M. (1990). The evolution of sex-change mechanisms in fishes. *Env. Biol. Fishes* **29**:81-93.

123. Sapolsky, R.M. (1982). The endocrine stress-response and social status in the wild baboon. *Hormones and Behavior* **16**(3):279-292.

124. Sapolsky, R.M. (1983). Endocrine aspects of social instability in the olive baboon (Papio anubis). *Amer. J. Primatology* **5**(4):365-379.

125. Sapolsky, R.M. (1985). Stress-induced suppression of testicular function in the wild baboon: role of glucocorticoids. *Endocrinology* **116**(6):2273-2278.

126. Sapolsky, R.M. (1986). Stress-induced elevation of testosterone concentration in high ranking baboons: role of catecholamines. *Endocrinology* **118**(4):1630-1635.

127. Sapolsky, R.M., (1988). Individual differences and the stress response: studies of a wild primate. *Adv. Exp. Med. Biol.* **245**:399-411.

128. Sapolsky, R.M. (1990). Adrenocortical function, social rank and pesonality among wil baboons. *Biological Psychiatry* **28**(10):862-878.

129. Satterlee, D.G. and Johnson, W.A. (1988). Selection of Japanese quail for contrasting blood corticosterone response to immobilization. *Poult. Sci.* **67**(1):25-32.

130. Schaefer, K.M., and Rubin, D.A. (1985). Effect of pH on sex ratio in chichlids and a poeciliid (Teleostei). *Copeia* (1):233-235.

131. Schmalhausen, I.I. (1949). **Factors of Evolution**. Blakeston, Philadelphia.

132. Selye, H. (1936). A syndrome produced by diverse nocuous agents. *Nature* **138**(3479):32.

133. Selye, H. (1946). The general adaptation syndrome and the diseases of adaptation. *J. Clinical Endo.* 6(2):117-231.

134. Selye, H. (1973). Stress and aerospace medicine. *Aerospace Med.* **44**:190-193.

135. Sibbald, W.S., Short,A., Cohen, M.P., and Wilson, R.F. (1977). Variations in adrenocortical responsiveness during severe bacterial infections. *Ann. Surg.* **186**:29-33.

136. Siegel, H.S. (1980). Physiological stress in birds. *BioScience* **30**:529-534.

137. Slotkin, T.A., Seidler, F.J., Kavlock, R.J. and Gray, J.A. (1992) Fetal dexamethasone exposure accelerates development of renal function: relationship to dose, cell differentiation and growth inhibition. *J. Dev. Physiol.* **17(2)**:55-61.

138. Smith, G.T., Wingfield, J.C., and Veit, R.R. (1994). Adrenocortical response to stress in the common diving-petrel, Pelecanoides urinatrix. *Physiol. Zool.* **67**:526-537.

139. Somero, G.N. (1978). Temperature adaptation of enzymes: Biological optimization through structure-function compromises. *Ann. Rev. Syst.* **9**:1-29.

140. Somero, G.N. (1995). Proteins and temperature. *Ann. Rev. Physiol.* **57**:43-68.

141. Somero, G.N., and DeVries, A.L. (1967). Temperature tolerance of some antarctic fishes. *Science* **156**:257-258 (1967).

142. Stearns, S.C. (1989). The Evolutionary significance of phenotypic plasticity. *BioScience* **39**(7):436-445.

143. Stearns, S.C., and Koella, J.C. (1986). The evolution of phenotypic plasticity in life-history traits: Predictions of reaction norms for age and size at maturity. *Evolution* **40**:893-913.

144. Stibor, H. (1992). Predator induced life-history shifts in a freshwater cladoceran. *Oecologia* **92**:162-165.

145. Stokkan, K.A., Harvey, S., Klandorf, H., Unander, S., and Blix, A.S. (1985). Endocrine changes associated with fat deposition and mobilization in Svalbard ptarmigan (*Lagopus mutus hyperboreus*). *Gen. Comp. Endocrinol.* **58**:76-80.

146. Stokkan, K.-A., Sharp, P.J., and Unander, S. (1986). The annual breeding cycle of the high arctic Svalbard ptarmigan (*Lagopus mutus hyperboreus*). *Gen. Comp. Endocrinol.* **61**:446-451.

147. Stonehouse, B. (1953). The Emperor penguin Apternodytes forsteri. 1. Breeding behavior and development. *Scient. Rep. Falkland Isl. Depend. Surv.* 6:1-33.

148. Stonehouse, B. (1956). The King Penguin of South Georgia. *Nature* **178**:1324-1426.

149. Stonehouse, B. (1970). Adaptation in polar and subpolar penguins. In **Antarctic Ecology, Volume 1** (ed. M.W. Holdgate). Academic Press, New York. pp. 526-541.

150. Swendsen, J., Hammen, C., Heller, T., and Gitlin, M. (1995). Correlates of stress reactivity in patients with bipolar disorder. *Am. J. Psychiatry* **152**(5):795-797.

151. Swezey and Somero (1982). Polymerization thermodynamics and structural stabilities of skeletal muscle actins from vertebrates adapted to different temperatures and hydrostatic pressures. *Biochemistry* **21**:4496-4503.

152. Tam, W.H., Birkett, L., Makaran, R., Payson, P.D., Whitney, D.K., and Yu, C.K.-C. (1987). Modification of carbohydrate metabolism and liver vitellogenic function in brook trout (*Salvelinus fontinalis*) by exposure to low pH. *Can J. Fish. Aquatic Sci.* **44**: 630-635.

153. Tauber, M.J., Tauber, C.A., and Maski, S. (1985). **Seasonal Adaptations in Insects**. Oxford University Press. New York.

154. Trouchot, J.-P., and Duhamel-Jouve, A. (1980). Oxygen and carbon dioxide in marine intertidal environment: diurnal and tidal changes in rockpools. *Respirin Physiol.* **39**:241-254.

155. Turek, F.W. (1985). Circadian neural rhythms in mammals. *Ann. Rev. Physiol.* **47**:49-64.

156. Virgin, C.E., Jr., Ha, T.P., Packan, D.R., Tombaugh, G.C., Yang, S.H., Horner, H.C., and Sapolsky, R.M. (1991). Glucocorticoids inhibit glucose transport and glutamate uptake in hippocampal astrocytes: implications for glucocorticoid neurotoxicity. *J. Neurochem.* **57**(4):1422-1428.

157. Wang, S. and Edens, F.W. (1993). Heat-stress response of broiler cockerels to manipulation of the gonadal steroids, testosterone and estradiol. *Comp. Biochem. Physiol (B)* **106(3)**:629-633.

158. Weimerskirch, H., Stachl, J.C., and Jouventin, P. (1992). The breeding biology and population dynamics of King Penguins *Aptenodytes patagonica* on the Crozet Islands. *Ibis* **134**:107-117.

159. West-Eberhard, M.J. (1989). Phenotypic plasticity and the origins of diversity. *Annu. Rev. Ecolo. Syst.* **20**:249-278.

160. Whittow, G.C. (1992). Laysan Albatross. In **The Birds of North America** (A. Poole, P. Stettenheim and F. Gill, Eds.), Vol. 2, No. 66, 20 pp. Academy of Natural Sciences, Philadelphia.

161. Wilckens, T. (1995). Glucocorticoids and immune function: physiological relevance and pathogenic potential of hormonal dysfunction. *Trends in Pharmacol. Sci.* **16(6)**:193-197.

162. Wilkinson, L. (1990). **SYSTAT: The System for Statistics.** Evanston, IL: SYSTAT, Inc.

163. Wingfield, J.C. (1973). Pituitary - steroid interrelationships in the plaice, *Pleuronectes platessa* L. PhD Thesis. University College of North Wales, Bangor.

164. Wingfield, J. C. (1994). Modulation of the adrenocortical response to stress in birds. In **Perspectives in Comparative Endocrinology**. (K.G. Davey, R.E. Peter, and S.S. Tobe, eds), pp 520-528. National Research Council Canada, Ottawa.

165. Wingfield, J.C., Deviche, P., Sharbaugh, S., Astheimer, L.B., Holberton, R., Suydam, R., and Hunt, K. (1994a). Seasonal changes of the adrenocortical responses to stress in Redpolls, *Acanthis flammea*, in Alaska. *J. Exp. Zool.* **270**:372-380.

166. Wingfield, J.C., and Farner, D.S. (1975). The determination of five steroids in avian plasma by radioimmunoassay and competitive protein binding. *Steroids* **26**:311-327.

167. Wingfield, J.C. and Farner, D.S. (1978a). The endocrinology of a naturally breeding population of the white-crowned sparrow (*Zonotrichia leucophrys pugetensis*). *Physiol. Zool.* **51**:188-205.

168. Wingfield, J.C. and Farner, D.S. (1978b). The annual cycle in plasma irLH and steroid hormones in feral populations of the white-crowned sparrow, *Zonotrichia leucophrys gambelii. Biol. Reprod.* **19**:1046-1056.

169. Wingfield, J.C., and Farner, D.S. (1979). Some endocrine correlates of renesting after loss of clutch or brood in the white crowned sparrow, *Zonotrichia leucophrus gambelli. Gen. Comp. Endo.* **38**:322-331.

170. Wingfield, J.C. and Hahn, T.P. (1994). Testosterone and territorial behavior in sedentary and migratory sparrows. *Anim. Behav.* **47**:77-89.

171. Wingfield, J.C., Hahn, T.P. and Doak, D. (1993). Integration of environmental cues regulating transitions of physiological state, morphology and behavior. In **Avian Endocrinology** (P.J. Sharp ed.), pp. 111-122, J. Endocrinology Ltd. Bristol.

172. Wingfield, J.C., Hahn, T.P., Levin, R., and Honey, P. (1992). Environmental predictability and control of gonadal cycles in birds. In **Biology of the Chordate Testis** (H. Grier and R. Cochran eds.), *J. Exp. Zool. Lond. 261*:214-231.

173. Wingfield, J.C., Hegner, R.E., and Lewis, D. (1991). Circulating levels of luteinizing hormone and steroid hormones in relation to social status in the cooperatively breeding white-browed sparrow weaver, *Plocepasser mahali. J. Zool. (London)* **225**:43-58.

174. Wingfield, J.C., Hillgarth, N., and Jacobs, J. (1996). Ecological constraints and the evolution of hormone-behavior interrelationships. *Proc. N.Y. Acad. Sci.* (in press).

175. Wingfield, J.C., and Kenagy, G.J. (1991), Natural regulation of reproductive cycles. In **Vertebrate Endocrinology: Fundamentals and Biomedical Implications, Volume 4B.** Academic Press, Inc. New York, New York.

176. Wingfield, J.C., and Monk, D. (1994). Behavioral and hormonal responses of male song sparrows to estrogenized females during the non-breeding season. *Horm. Behav.* **28**:146-154.

177. Wingfield, J.C., Newman, A., Hunt, G.L.Jr., and Farner, D.S. (1982). Endocrine aspects of female-female pairing in the western gull (*Larus occidentalis wymani*). *Anim. Behav.* **30**:9-22.

178. Wingfield, J.C., Schwabl, H., and Mattocks, P.W.Jr. (1990). Endocrine mechanisms of migration. In **Bird Migration** (E. Gwinner ed.), pp. 232-256, Springer-Verlag, Berlin.

179. Wingfield, J.C., Smith, J.P., and Farner, D.S. (1982). Endocrine response of white-crowned sparrows to environmental stress. *Condor* **84**:399-409.

180. Wingfield, J.C., Suydam, R., and Hunt, K. (1994b). The adrenocortical responses to stress in Snow Buntings (*Plectrophenax nivalis*) and Lapland Longspurs (*Calcarius lapponicus*) at Barrow Alaska. *Comp. Biochem. Physiol.* **108**:299-306.

181. Wingfield, J.C., Vleck, C.M., and Moore, M.C. (1992). Seasonal changes of the adrenocortical response to stress in birds of the Sonoran desert. *J. Exp. Zool.* **264**:419-428.

182. Wise, P.M., and Frye, B.E. (1973). Functional development of the hypothalamo-hypophyseal-adrenal cortex axis in the chick embryo, *Gallus domesticus*. *J. Exp. Zool.* **185**:277-292.

183. Wyndham, E. (1986). Length of birds' breeding seasons. *Am. Nat.* **128**:155-164.

184. Zanchetti, A. (1972). Expectancy and the pituitary-adrenal system: Discussion in: **Physiology, Emotion and Psychosomatic Illness. Ciba Found. Symp. 8**. Elsevier/North-Holland, Masterdam, pp 296.

185. Zann, R.A., Morton, S.R., Jones, K.R., and Burley, N.T. (1995). The timing of breeding by zebra finches in relation to rainfall in central Australia. *Emu* **95**:208-222.

APPENDIX A. - PREDICT: A program to calculate the predictability of an environment, and the portions of that predictability due to constancy and contingency.

1. General Description

This program calculates the predictability for an environment and the two components of predictabiliy, constancy and contingency, using the methods of Colwell (1974). It was designed to compile on Macintoshes (using Think-C), IBM-PCs (using Borland-C), and UNIX (using GNU-C) machines. Data sets can either be entered line-by-line from as the program prompts (this is the only way on the MAC), or a filename can be supplied on the command line. The data is entered with data separated by commas, with the time intervals in columns, and the classes in rows.

2. Source Listing

```
/*
    *-------------------------------------------------------------*
    | This program determines the Predictablity, Constancy and    |
    | Contingency of Periodic Phenomena as described by Robert Colwell |
    | in Ecology 55:1148-1153(1974).                              |
    |                                                             |
    | This version is designed to compile under DOS, UNIX, and on a |
    | Macintosh using Think-C. The use of K & R procedure notation |
    | was required for the old version of the Think-C compiler.   |
    *-------------------------------------------------------------*
*/
#include <stdio.h>
#include <ctype.h>
#include <math.h>

#ifdef __MSDOS__                    /*  We are on a real computer.  */
#       include <stdlib.h>          /*  DOS lives!!                 */
#       include <process.h>
#       include <alloc.h>
#       include <string.h>
#   else
#       ifdef unix || __unix__      /* We are on a unix machine. */
#           include <stdlib.h>
#           include <string.h>
#       else
#           define __MAC__          /* Not a real computer, so     */
#           include <storage.h>     /* assume it is a Macintosh.   */
#           include <unix.h>
#           include <strings.h>
#       endif
#endif

/* *-------------------------------* */
/* |     Define Constants          | */
/* *-------------------------------* */
#define TRUE        1
#define FALSE       0
#define EOD         0x1A
```

```
/* *-------------------------------------* */
/* |          Macros                     | */
/* *-------------------------------------* */
#define IS_NUMERIC(c) ( (((c)>='0' && (c))<='9' ? 1:0 ) + ((char)(c) == '.' ? 1:0 ) )
#define IS_NOT_ZERO(c) ( ( ( (c) > 1 ) || (int)( 100000.0 * (c) )) ? TRUE : FALSE )

/* *-------------------------------------* */
/* |      Subroutine Prototypes          | */
/* *-------------------------------------* */
int       getdata( void );
int       finished( char * );
float *   tokenize( char * );
void      calc_predictability( void );

/* +-------------------------+ */
/* | Global variables        | */
/* +-------------------------+ */
float **N;
float X[100], Y[100];
int       States = 1;
int       Times_Per_Cycle;
int       Times = 0;
int       FLOAT_POINTER_SIZE = sizeof( float * );
char    *version="Version 2.0\n\n";
char    *copyright="Nudibranch Central Software 1993-1996 (Jerry Jacobs)";


#ifdef __MAC__
int
main() {
#else
int
main( int argc, char*argv[] ) {
#endif
    int done=FALSE;

    puts("Colwell Predictability Model");
    puts( version );
    puts("Nudibranch Central Software 1993-1996\n");

#ifndef __MAC__
    switch ( argc ) {
        case 1:
#endif
            while ( !done ) {
                done = getdata();
                if ( !done ) {
                    calc_predictability();
                    puts("Press ENTER to continue.");
                    (void)getc( stdin );
                    puts("\n");
                }
            }
#ifndef __MAC__
            break;
        case 2:
            if ( read_data( argv[1] ) )
                calc_predictability();
            break;
```

```
            default:
                puts( 'Too many command line arguments.' );
                puts( 'SYNTAX - predict');
                puts( '        or  predict filename' );
                puts( '\n\n' );
                return 1;
        }
#endif
        puts("Done.\n" );
        return(0);
}

void
calc_predictability( ) {
        int i, j, k;
        float HX=0, HY=0;
        float HXY=0;
        float Z=0;
        float *_STATE_;
        float ConditionalUncertainty;
        float Predictability;
        float C, M, Gm, Gc, Gp;
        float S;
        char answer[10];
        char filename[80];
        char tofile=FALSE;
        FILE *output, *stream;
        char fileopen;

        S = States;
        /* *-------------------------* */
        /* | Initialize Arrays.      | */
        /* *-------------------------* */
        for (j=0; j<Times; j++ ) {
                X[j] = 0.0;
        }
        for (i=0; i<States; i++ ) {
                Y[i] = 0.0;
        }
        /* *-----------------------------------------* */
        /* | Determine Column Totals and Grand Total | */
        /* *-----------------------------------------* */
        for (j=0; j<Times; j++ ) {
                for (i=0; i<States; i++ ) {
                        _STATE_ = N[i];
                        _STATE_++;
                        X[j] = X[j] + _STATE_[j];
                }
                Z = Z + X[j];
        }
        /* *-------------------------* */
        /* | Determine Row Totals    | */
        /* *-------------------------* */
        for (i=0; i<States; i++ ) {
                _STATE_ = N[i];
                _STATE_++;
                for (j=0; j<Times; j++ ) {
                        Y[i] = Y[i] + _STATE_[j];
                }
        }
```

```
/* *-------------------------------------------* */
/* | Determine Uncertainty with respect to time | */
/* *-------------------------------------------* */
for (j=0; j<Times; j++ ) {
      if ( IS_NOT_ZERO( X[j] ) )
            HX = HX - ( X[j]/Z * log( X[j] / Z ) );
}
/* *-------------------------------------------* */
/* | Determine Uncertainty with respect to state. | */
/* *-------------------------------------------* */
for (i=0; i<States; i++ ) {
      if ( IS_NOT_ZERO( Y[i] ) )
            HY = HY - ( Y[i]/Z * log( Y[i] / Z ) );
}
/* *-------------------------------------------------------* */
/* | Determine Uncertainty with respect to interaction of time | */
/* | and state.                                              | */
/* *-------------------------------------------------------* */
for (j=0; j<Times; j++ ) {
      for (i=0; i<States; i++ ) {
            _STATE_ = N[i];
            _STATE_++;
            if ( IS_NOT_ZERO( _STATE_[j] ) ) {
                  HXY = HXY - ( _STATE_[j]/Z * log( _STATE_[j] / Z ) );
            }
      }
}

ConditionalUncertainty = HXY - HX;
Predictability = 1 - ( ConditionalUncertainty / log( S ) );

/* *-----------------------* */
/* | Calculate Constancy | */
/* *-----------------------* */
C = 1 - ( HY / log( S ) );
/* *-----------------------* */
/* | Determine Contingency | */
/* *-----------------------* */
M = (HX + HY - HXY ) / log( S );

printf( 'Send output to a file (as well as the screen)? ' );
gets( answer );
if (strlen(answer) != 0 )
        {
            if ( (*answer == 'y') || (*answer == 'Y') )
                {
                    fileopen = FALSE;
                    while ( !fileopen ) {
                        printf( 'Filename: ' );
                        gets( filename );
                        if ( (output=fopen( filename, 'r' )) != NULL ) {
                            fclose( output );
                            printf( 'File already exists.  Overwrite? ' );
                            gets( answer );
                            if (strlen(answer) == 0 )
                                continue;
                            if ( (*answer == 'n') || (*answer == 'n') )
                                continue;
                        }
                        if ( (output=fopen( filename, 'w' )) == NULL ) {
                            printf('Error trying to open output file %s.\n', filename );
```

```
                                exit (1);
                            }
                            tofile = TRUE;
                            fileopen=TRUE;
                        }
                        tofile = TRUE;
                    }
                else
                    tofile = FALSE;
            }
        else
            tofile = FALSE;
    Gm = 2 * Z * ( HX + HY + HXY );
    Gc = 2 * Z * ( log( S ) - HY );
    Gp = Gm + Gc;
    for (k=0; k<1+tofile; k++ ) {
        if ( tofile && k==1 )
            stream = output;
         else
            stream = stdout;
        if ( Times > 15 )
                fprintf( stream, 'Data matrix too large to display. \n' );
            else
                {
                    fprintf( stream,'Input Data:\n' );
                    fprintf( stream,'----------\n' );
                    for (i=0; i<States; i++ ) {
                        _STATE_ = N[i];
                        for (j=0; j<Times; j++ ) {
                            fprintf( stream,'%5.1f ', _STATE_[j+1] );
                        }
                        fprintf( stream,'\n' );
                    }
                    fprintf( stream,'\n' );
                }
        fprintf( stream,'H(XY) = %11.5f\n', HXY);
        fprintf( stream,'H(X)  = %11.5f\n', HX);
        fprintf( stream,'H(Y)  = %11.5f\n', HY);
        fprintf( stream,'Log(s)= %11.5f\n', log( S ) );
        fprintf( stream,'Grand Total (Z) = %7.1f\n', Z );
        fprintf( stream,'Predictability   = %9.5f\n', fabs( Predictability ) );
        fprintf( stream,'Constancy (C)    = %9.5f\n', fabs( C ) );
        fprintf( stream,'Contingency (M)  = %9.5f\n', fabs( M ) );
        fprintf( stream,'\n" );
        fprintf( stream,'G statistics for:\n' );
        fprintf( stream,' Contingency (Gm)    = %11.5f\n', ( IS_NOT_ZERO(Gm) ?  Gm : 0.0 ) );
        fprintf( stream,' Contstancy (Gc)     = %11.5f\n',  ( IS_NOT_ZERO(Gc) ? Gc : 0.0 ) );
        fprintf( stream,' Predictability (Gp) = %11.5f\n',  ( IS_NOT_ZERO(Gp) ? Gp : 0.0 ) );
        fprintf( stream,'\n' );
        if (k == 1 )
            fclose( output );
    }
    return;
}

/* Read Data from a File */
int
read_data( filename )
    char *filename;
{
    char input[255];
```

```
    int T,Max_Times=0;
    float *tokenized;
    int first=TRUE;
    FILE *dataFile;

    States  = 1;
    Times   = 0;
    T       = 0;
    if ( (dataFile = fopen( filename, 'r' ) ) == NULL ) {
        printf( 'Unable to open data file: %s\n', filename );
        return 0;
    }

    while ( fgets(input, 254, dataFile) != NULL ){
        tokenized = tokenize( input );
        if ( !finished( input ) ) {
            T = tokenized[0];
            if (((T>Max_Times) && !first) || T<Max_Times ) {
                    if ( T<Max_Times )
                        printf('\a—Error - This line has less times!\n');
                      else
                    printf('\a—Error - This line has more times!\n');
                }
              else
                {
                    if ( first ) {
                            if ( ( N = (float **)malloc( FLOAT_POINTER_SIZE )) == NULL) {
                                printf('Error allocating memory\n');
                                exit (1);
                            }
                            Max_Times = T;
                            first = FALSE;
                        }
                      else {
                            if ((N = (float **)realloc( N, FLOAT_POINTER_SIZE * (States + 1 ))) == NULL) {
                                printf('Error reallocating memory\n');
                                exit (1);
                            }
                        }
                    N[States-1]=tokenized;
                    States++;
                }
        }
    } /* Done getting input */
    States--;
    Times = T;
    if ( (States == 0) || (Times == 0) )
            return (0);
        else
            return (1);
}

int
getdata() {
    char input[255];
    int done=FALSE;
    int T,Max_Times=0;
    float *tokenized;
    int first=TRUE;
    static char border[67];
```

```
printf( "\n" );
printf( "Enter the rows of the matrix with each datum separated by a " );
printf( "comma.\n" );
printf( "When done, leave a blank line.\n" );
memset( border, '-', 66 );
border[65] = 0 ;
printf( "%s\n", border );
printf( "\n" );

States      = 1;
Times       = 0;
T           = 0;

while ( !done ){
     tokenized = 0;
     while ( !tokenized ) {
          printf("State %2.2u: ",States);
          gets( input );
          tokenized = tokenize( input );
     }
     if ( finished( input ) )
               done = TRUE;
          else
               {
                    T = tokenized[0];
                    if (((T>Max_Times) && !first) || T<Max_Times )
                         {
                              if ( T<Max_Times )
                                   printf("\a---Error - This line has less times!\n" );
                                 else
                                   printf("\a---Error - This line has more times!\n" );
                         }
                    else
                         {
                              if ( first )
                                   {
                                        if ( ( N = (float **)malloc( FLOAT_POINTER_SIZE )) == NULL)
                                        {
                                             printf("Error allocating memory\n" );
                                             exit (1);
                                        }
                                        Max_Times = T;
                                        first = FALSE;
                                   }
                                 else
                                   {
                                        if ((N = (float **)realloc( N, FLOAT_POINTER_SIZE * (States + 1 ))) ==
                                             NULL)
                                        {
                                             printf("Error reallocating memory\n" );
                                             exit (1);
                                        }
                                   }
                              N[States-1]=tokenized;
                              States++;
                         }
               }
} /* Done getting input */
States--;
Times = T;
```

```
    if ( (States == 0) || (Times == 0) )
            return (TRUE);
        else
            return (FALSE);
}

int
finished( input )
    char *input;
{
    if ( *input == EOD )
            return ( TRUE );
        else
            return ( FALSE );
}

char drbottom[]={   0x0A, 0x44, 0x72, 0x2E, 0x20, 0x47, 0x2E, 0x4A,
                    0x2E, 0x20, 0x4B, 0x65, 0x6E, 0x61, 0x67, 0x79,
                    0x20, 0x69, 0x73, 0x20, 0x61, 0x20, 0x68, 0x6F,
                    0x72, 0x73, 0x65, 0x27, 0x73, 0x20, 0x61, 0x73,
                    0x73, 0x21, 0x21, 0x21, 0x0A, 0x00 };

float *
tokenize( input )
    char *input;
{
    char *token, *next_token;
    float token_count=0;
    int t_count;
    char *begin;
    float *tokenized;
    int i;
    int length;
    char scratch[255];

    strcpy( scratch, input );
    if ( strlen( input ) == 0 ) {
        *input=EOD;
        return( (float *)EOD );
    }
    begin = input;
    /* *-------------------* */
    /* | Search for first digit | */
    /* *-------------------* */
    for (i=0; i<strlen(input); i++ ) {
        if ( IS_NUMERIC( *begin ) )
            break;
        begin++;
    }
    /* *-----------------------------* */
    /* | If no digits found signal done | */
    /* *-----------------------------* */
    if ( i == strlen( input ) ) {
        *input=EOD;
        return( (float *)EOD );
    }
    token = begin;
    /* *-------------------* */
    /* | Search for commas. | */
    /* *-------------------* */
    while ((next_token = strchr(token, ',')) != NULL ) {
```

```
        token_count++;
        /* *-------------------------* */
        /* | Put a 0 at end of number. | */
        /* *-------------------------* */
        for (i=0; i<strlen(token); i++ ) {
            if (!isdigit( token[i] ) && token[i] != '.' ) {
                token[i] = 0;
                break;
            }
        }
        token = next_token;
        token++;
        /* *-------------------------------* */
        /* | Move to the start of the next token | */
        /* *-------------------------------* */
        length = strlen(token);
        for (i=0; i<length; i++ ) {
            if ( *token == 0 ) {
                printf(" Error in input.\n" );
                printf("    [1]\n" );
                return ( NULL );
            }
            if ( IS_NUMERIC( *token ) )
                break;
            token++;
        }
    }
    /* *-------------------------------* */
    /* | Search for beginning of last token | */
    /* *-------------------------------* */
    for (i=0; i<strlen(token); i++ ) {
        if ( !IS_NUMERIC( token[i] ) ) {
            token[i] = 0;
            break;
        }
    }
    token_count++;
    t_count = token_count + 1;
    if ((tokenized = (float *)malloc( ( t_count ) * sizeof( float ) )) == NULL ) {
        printf("\nMemory allocation error.\n" );
        exit (255);
    }
    tokenized[0] = token_count;
    token_count=0;

    begin = scratch ;
    /* *-----------------* */
    /* | Search for first digit | */
    /* *-----------------* */
    for (i=0; i<strlen(scratch); i++ ) {
        if ( IS_NUMERIC( *begin ) )
            break;
        begin++;
    }
    token = begin;

    /* *-----------------* */
    /* | Search for commas. | */
    /* *-----------------* */
    while ((next_token = strchr(token, ',')) != NULL ) {
```

```
        /* *-------------------------* */
        /* | Put a 0 at end of number. | */
        /* *-------------------------* */
        for (i=0; i<strlen(token); i++ ) {
            if (!isdigit( token[i] ) && token[i] != '.' ) {
                token[i] = 0;
                break;
            }
        }
        t_count = (int)token_count+1;
        tokenized[t_count] = atof( token );
        token_count++;
        token = next_token;
        token++;
        /* *-------------------------------* */
        /* | Move to the start of the next token | */
        /* *-------------------------------* */
        length = strlen(token);
        for (i=0; i<length; i++ ) {
            if ( *token == 0 ) {
                printf(" Error in input.\n" );
                printf("    [4]\n" );
                return ( NULL );
            }
            if ( IS_NUMERIC( *token ) )
                break;
            token++;
        }
    }
    t_count = (int)token_count+1;
    tokenized[t_count] = atof( token );
    return ( tokenized );
}
/* *-----------------------------------------------------* */
/* |                End of Predict.C                     | */
/* *-----------------------------------------------------* */
```

3. Sample Input

The following data for precipitation patterns at Uaupes, Brazil (1951-1960) comes from Colwell (1974). Each column represents a month (Jan-Dec), and the data in the rows represent the number of times each *rainfall state* (or class) occurred during that month. The *rainfall states* were on a $\log_2$ scale, with state 1 assigned to no measurable precipitation.

Rainfall State	J	F	M	A	M	J	J	A	S	O	N	D
1												
2												
3												
4												
5												
6												
7												
8												
9							1		3	2		
10	6	5	4	6	1	4	4	10	7	6	7	3
11	4	5	6	4	9	6	5			2	3	7
12												

Data file used as input for PREDICT.EXE using the above data:

```
0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,1,0,3,2,0,0
6,5,4,6,1,4,4,10,7,6,7,3
4,5,6,4,9,6,5,0,0,2,3,7
0,0,0,0,0,0,0,0,0,0,0,0
```

4. Sample Output

Colwell Predictability Model
Version 2.0

Nudibranch Central Software 1993-1996

Send output to a file (as well as the screen)? n
Input Data:

0.0	0.0	0.0	0.0	.0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	3.0	2.0	0.0	0.0
6.0	5.0	4.0	6.0	1.0	4.0	4.0	10.0	7.0	6.0	7.0	3.0
4.0	5.0	6.0	4.0	9.0	6.0	5.0	0.0	0.0	2.0	3.0	7.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

H(XY) - 3.10461
H(X) - 2.48491
H(Y) - 0.85173
Log(s) - 2.48491
Grand Total (Z) - 120.0
Predictability - 0.75061
Constancy (C) - 0.65724
Contingency (M) - 0.09337

G statistics for:
Contingency (Gm) - 1545.90063
Contstancy (Gc) - 391.96188
Predictability (Gp) - 1937.86255

Done.

APPENDIX B. Radioimmunoassay Calculation Program

A. General Description

The RIA calculation program is used to capture data from a Beckman LC3801 and perform calculations on the data to determine hormone concentrations in blood samples. The scintillation counter calculates the actual raw dose, and the program integrates recoveries and plasma volumes to determine final plasma concentrations.

B. Shared Procedures (RIALIB.LIB)

1. Header Files

a. Header file for Lotus 1-2-3 reading procedures.

```
// *-------------------------------------------*
// | Contains constants for getting information from Lotus |
// | 1-2-3 files.                              |
// *-------------------------------------------*
#ifndef _123_
#define _123_


#define FILE_START      0
#define BLANK_CELL      12
#define INTEGER         13
#define REAL_NUMBER     14
#define LABEL           15
#define FORMULA         16
#define READERROR       -1

extern char *datapath;
#endif
```

b. Header file for RIA Library.

```
// *-------------------------------------------*
// | FILE: RIALIB.H                            |
// |-------------------------------------------|
// | Header file for the RIA library           |
// *-------------------------------------------*
#ifndef _RIALIB_H_
#define _RIALIB_H_
#include <conio.h>
#include <stdio.h>
/* *-------------------------------------* */
/* |  -     Global Constants             | */
/* *-------------------------------------* */
#define TRUE        -1
#define FALSE       0
#define RIGHT       77
#define LEFT        75
#define DELETE      83
#define HOME        71
#define END         79
#define CR          13
#define LF          10
#define F2          60
#define RETURN      13
#define BACKSPACE   8
#define ESC         27
#define CNTRL_C     3
#define INPUT_OK    0
#define INPUT_BAD   -1
#define IGNORE      0
#define RETRY       1
#define ABORT       2
/* *-------------------------------------* */
/* |  Data from Prompt for Disk          | */
/* *-------------------------------------* */
#define RESET_DRIVE         0
#define VERIFY_SECTORS      4
#define HEAD                0
#define DISK                0
#define TRACK               0
#define SECTOR              1
#define NUMBER_SECTORS      1

/* *-------------------------------------* */
/* |  Type definitions.                  | */
/* *-------------------------------------* */
typedef struct Window_Style {
        int forecolor;
        int backcolor;
        int textcolor;
        int x1;
        int y1;
        int x2;
        int y2;
        int titlefore;
        int titleback;
        char title[81];
} WINSTYLE;

extern struct text_info TEXT;

/* *-------------------------------------* */
/* |  Funtion Prototypes                 | */
/* *-------------------------------------* */
char *allocate ( int );
void Beep( void );
void Center_Text( char * msg, struct Window_Style Window );
int Copy_File( FILE *File, char *Filename, char *datapath, int
        *Copied_Files, WINSTYLE *File_Window, WINSTYLE
        *Records_Window, char Show_Data[][12] );
int draw_box( int, int, int, int );
void die (char *);
extern int  Display_Critical_Error( char * );
FILE *Get_File( char *Filename, char *Drive, int *start );
void Get_User_Name( struct Window_Style *Name_Window,
        char *User_Name, char *Time_Stamp );
```

```
void far Hardware_Error( unsigned deverr, unsigned errval,
        unsigned far *devhdr );
int Input_String( char *string, int width, int position, int
        describe );
int Overwrite( char *Filename );
void Prompt_New_Disk( char *Prompt_Disk, char *Drive );
void Read_Descriptions( char *datapath, int *fcount, char
        **file_desc );
void Save_Desc( char *datapath, char **file_desc, char
        **new_desc, int Copied_Files, int fcount );
void Screen_Setup( struct Window_Style ** );
int strdel( char *, int );
int strins( char *, char, int, int );
char *strstore ( char *string );
void Window_Box( struct Window_Style * );
void Window_Restore( struct Window_Style *Window, char
        *savebuffer );
char *Window_Save( struct Window_Style *Window );

#endif
```

2. *RIA Library Procedure Sources.*

a. Lotus 1-2-3 Data File Title Reading Procedure

```
//*-------------------------------------------------------*
//| SOURCE FILE: 123.C                                    |
//*-------------------------------------------------------*
//| Routines to read title information from Lotus 1-2-3 data files. |
//|                                                       |
//| Contains procedures:    read_int    Reads an integer from |
//|                                     a 123 file.       |
//|                         get_format  Reads the format and |
//|                                     location for a cell. |
//|                         get_123_title Reads the title from |
//|                                     an RIA 123 data file. |
//*-------------------------------------------------------*
#include <stdio.h>
#include <alloc.h>
#include <process.h>
#include <mem.h>
#include <string.h>
#include '123.h'


//*-------------------------------------------------------*
//| PROCEDURE: read_int                                   |
//*-------------------------------------------------------*
//| PURPOSE:   Reads a 2 byte integer from a 123 file.    |
//| ON CALL:   file    - Pointer to structure of opened 123 |
//|                      data file.                       |
//|            eof_code - Pointer to Flag indicating if   |
//|                       End-of-file reached.            |
//| RETURNS:   2 byte integer if OK.                      |
//|            READERROR if premature EOF.                |
//*-------------------------------------------------------*
int
read_int( FILE *file, int *eof_code ) {
    int hi,low;

    *eof_code = 0;
    if( (low = fgetc( file ))==EOF ) {
        *eof_code = 1;
        exit(1);
        return (READERROR);
    }
    if((hi  = fgetc( file )) ==EOF) {
        *eof_code = 1;
        exit(1);
        return (READERROR);
    }
    return ((hi << 8) + low );
}

//*-------------------------------------------------------*
```

```
//*-------------------------------------------------------*
//| PROCEDURE: get_format                                 |
//*-------------------------------------------------------*
//| PURPOSE:   Reads a cell format and location.          |
//| ON CALL:   file     - Pointer to structure of opened 123 |
//|                       data file.                      |
//|            format   - Pointer to format to return.   |
//|            column   - Pointer to cell column to return. |
//|            column   - Pointer to cell row to return.  |
//| RETURNS:   Nothing                                    |
//*-------------------------------------------------------*
void
get_format( FILE *file, int *format, int *column, int *row) {
    int i;

    if ((i=fgetc( file )) == EOF) {
        printf("Premature EOF\n");
        exit(99);
    }
    *format = i;
    *column = read_int( file, &i);
    *row    = read_int( file, &i);
}

//*-------------------------------------------------------*
//| PROCEDURE: get_123_title                              |
//*-------------------------------------------------------*
//| PURPOSE:   Extracts the title from a 123 RIA worksheet. |
//| ON CALL:   filename - Pointer to the file name to get the |
//|                      title from.                      |
//|            format - Pointer to format to return.      |
//|            column - Pointer to cell column to return. |
//|            column - Pointer to cell row to return.    |
//|                      End-of-file reached.             |
//| RETURNS:   Pointer to the worksheet title.            |
//*-------------------------------------------------------*
char *
get_123_title( char *filename ) {
    FILE *file;
    int done;
    int OPCODE, LENGTH/*,misc */;
    int format, column, row,i,j,k;
    char *Label;
    char * TMP;
    double Double;
    char Tmpdbl[8];
    char *EOFerror="premature end of file\n";

    if ((file = fopen( filename,'rb' )) == NULL ) {
        printf("Failed to open %s in module
    123.c.\n",filename);
        return ( NULL );
    }
    while ((OPCODE = read_int( file, &done ) ) != 1 ) {
```

```
if (!done)
    {
        LENGTH = read_int( file, &done );
        switch ( OPCODE ) {
            case FILE_START:
                read_int( file, &done );
                break;
            case BLANK_CELL:
                get_format( file, &format, &row,
                    &column );
                if ( ( row | column) == 0 ){
                    fclose( file );
                    return( NULL );
                }
                break;
            case INTEGER:
                get_format( file, &format, &row,
                    &column );
                read_int( file, &done );
                if (( row | column) == 0 ){
                    fclose( file );
                    return( NULL );
                }
                break;
            case REAL_NUMBER:
                TMP = (char *)&Double;
                get_format( file, &format, &row,
                    &column );
                for (i=0; i<8; i++ ) {
                    if ((j = fgetc( file ))== EOF) {
                        fclose( file );
                        return ( NULL );
                    }
                    Tmpdbl[i]=j;
                }
                memcpy( TMP, Tmpdbl,8);
                if ( (row | column) == 0 ){
                    fclose( file );
                    return ( NULL );
                }
                break;
            case LABEL:
                get_format( file, &format, &row,
                    &column );
                if (( Label = (char *) malloc(
                    LENGTH - 6 +2 ))==NULL ){
                    printf("Malloc error.\n");
                    fclose( file );
                    return ( NULL );
                }
                k=0;
                for (i=0; i<= (LENGTH - 6); i++ ) {
                    if ( (j = fgetc( file )) == EOF )
                        exit(0);
                    if (i != 0 && j !='\'' && j != '\"' &&
                        j != '^' ) {
                        Label[k++]=j;
                    }
                }
                Label[i] = 0;
                if ( (row | column) == 0 ){
                    fclose( file );
                    return ( Label );
                }
                break;
            case FORMULA:
            default:
                for (i=0; i<LENGTH; i++ ) {
                    if ((j = fgetc( file ))== EOF) {
                        fclose( file );
                        puts( EOFerror );
                        return ( NULL );
                    }
                }
                break;
            }
        }
        else
            {
                printf("Premature end of file.\n");
                fclose( file );
                return( NULL );
            }
    }
    fclose( file );
    return ( NULL );
}
```

b. Memory Allocation Procedure

```
// *-------------------------------------------------*
// | SOURCE FILE: ALLOCAT.C                          |
// *-------------------------------------------------*
// | Memory allocation routine used for RIA program. |
// *-------------------------------------------------*
#include "rialib.h"
#include <alloc.h>

// *-------------------------------------------------------------*
// | PROCEDURE: allocate                                         |
// *-------------------------------------------------------------*
// | PURPOSE:   Dynamically allocates memory from the            |
// |            global heap                                      |
// | ON CALL:   block_size - Size of the desired memory block.   |
// | RETURNS:   Pointer to allocated block.                      |
// |                If not enough memory, displays error,        |
// |                    then exits to system                     |
// *-------------------------------------------------------------*
char *
allocate ( int block_size ) {
    char *temp;

    if (( temp = (char*) malloc( block_size )) == NULL )
    die("Ran out of memory during \'malloc\' call.\n");
    return ( temp );
}
```

c. Alarm Procedure

```
// *-------------------------------*
// | SOURCE FILE: BEEP.C           |
// |-------------------------------|
// | Alarm noise for RIA program.  |
// *-------------------------------*
#include <dos.h>

// *-------------------------------------------*
// | PROCEDURE: Beep                           |
// *-------------------------------------------*
// | PURPOSE:   Sounds a warning beep at 110 Hertz. |
// | RETURNS:   Nothing.                       |
// *-------------------------------------------*
void
Beep() {
    sound(110);
```

```
        delay(200);
        nosound();
}
```

d. Text Centering Procedure

```
// *-------------------------------------------*
// | SOURCE FILE: CENTTEXT.C                   |
// *-------------------------------------------*
// | Routine to center text in a window.       |
// *-------------------------------------------*
#include 'rialib.h'
#include <string.h>

// *-------------------------------------------------------------*
// | PROCEDURE: Center_Text                                      |
// *-------------------------------------------------------------*
// | PURPOSE:   Center a text message within a window.           |
// | ON CALL:   msg   - Pointer to message to display.           |
// |                    Window - Structure defining window.      |
// | RETURNS:   Nothing.                                         |
// *-------------------------------------------------------------*
void
Center_Text( char * msg, struct Window_Style Window ) {
        int x,y;
        y = wherey();
        x = ( (Window.x2 - Window.x1 +1 ) - strlen(msg) ) / 2 +
                Window.x1;
        gotoxy(x,y);
        cputs( msg );
}
```

e. File Copying Procedure

```
// *-------------------------------------------*
// | SOURCE FILE: COPYFILE.C                   |
// *-------------------------------------------*
// | Routine to copy a file from the 5-1/4' Floppy Drive. |
// *-------------------------------------------*
#include <stdio.h>
#include 'rialib.h'
#include <string.h>

// *-------------------------------------------------------------*
// | PROCEDURE: Copy_File                                        |
// *-------------------------------------------------------------*
// | PURPOSE:                                                    |
// | ON CALL: File          - Pointer to structure for opened    |
// |                          file to copy.                      |
// |          Filename      - Name of file opened (no path).     |
// |          datapath      - Path to RIADATA directory.         |
// |          Copied_Files  - Number of files copied.            |
// |          File_Window   - Definition for Filename Window.    |
// |          Records_Windows - Definition for Records Dumped    |
// |                          Window.                            |
// |          Show_Data     - Array for first 12 records.        |
// | RETURNS: If Copy OK then, then returns number of records.   |
// |          If no records in file, then returns -1.            |
// |          If error, then returns 0.                          |
// *-------------------------------------------------------------*
int
Copy_File( FILE *File, char *Filename, char *datapath,
           int *Copied_Files, WINSTYLE *File_Window,
           WINSTYLE *Records_Window,
           char Show_Data[][12] ) {
    char Input_Line[80];
    char Output_Filename[80];
    FILE *Output_File;
    int Records=0, Line=0,i;
    char Error_Msg[100];

    strcpy( Output_Filename, datapath );
    strcat( Output_Filename, Filename );
    //
    // Try to open the file to see if it already exists.
    //
    if (( Output_File = fopen ( Output_Filename,'r')) != NULL )
    {
        /* File already exists */
        if ( !Overwrite( Output_Filename ) ) {
            fclose( Output_File );
            return 0;
        }
    }
    if (( Output_File = fopen ( Output_Filename,'w')) == NULL
    ) {
        /* Can't open output file. */
        sprintf( Error_Msg,'Can\'t open %s for writing.\n',
    Output_Filename);
        die( Error_Msg );
    }
    while ( ( fgets( Input_Line, 80, File ) ) != NULL ) {
    if ( fputs( Input_Line, Output_File ) == EOF ) {
        /* Error during write */
        sprintf( Error_Msg,'Error during write to %s\n',
            Output_Filename);
        die( Error_Msg );
    }
    ++Records;
    if (Records < 12 ) {
        strncpy(&Show_Data[Records-1][0],Input_Line,10);
        for ( i=0; i<10; i++ ) {
            if ( Show_Data[Records-1][i] == 10 ||
                 Show_Data[Records-1][i] == 13 ) {
                 Show_Data[Records-1][i] =0;
                 i = 100;
            }
        }
    }
    }
    ++(*Copied_Files);
    window( File_Window->x1+1, File_Window->y1+1,
    File_Window->x2-1, File_Window->y2-1);
    textcolor( File_Window->textcolor);
    textbackground( File_Window->backcolor);
    Line = *Copied_Files;
    if (*Copied_Files >
            (File_Window->y2 - File_Window->y1 - 1 ) )
    {
        Line = File_Window->y2 - File_Window->y1 - 1;
        gotoxy( 1,Line );
        cputs("\r\n");
    }
    gotoxy( 1, Line );
    cprintf( '%s',Filename );
    window( Records_Window->x1+1, Records_Window->y1+1,
    Records_Window->x2-1, Records_Window->y2-1);
    if ( *Copied_Files >
            (File_Window->y2 - File_Window->y1 - 1 ) )
    {
        gotoxy( 1, Line );
        cputs("\r\n");
    }
    gotoxy( 1, Line );
    cprintf( '  %d',Records );
    window( 1,1,80,24);
```

```
        fclose( File );
        fclose( Output_File );
        if ( Records == 0 )
                return (-1);
            else
                return ( Records );
}
```

f. Error Exit Procedure

```
// |-------------------------------------|
// | SOURCE FILE: DIE.C                  |
// |-------------------------------------|
// | Routine to display error and Exit.  |
// |-------------------------------------|
#include <conio.h>
#include <stdio.h>
#include <process.h>
#include <ctype.h>

extern struct text_info TEXT={0,0,0,0,0,0,0,0,0,0,0 };

// *-------------------------------------------------*
// | PROCEDURE: die                                  |
// *-------------------------------------------------*
// | PURPOSE: Display an error message and exit to system |
// | ON CALL: Message - points to the message to print.   |
// | RETURNS: Nothing.                                    |
// *-------------------------------------------------*

void
        die (char *Message ) {
        textattr( TEXT.attribute );
        clrscr();
        fprintf(stderr, Message);
        exit(1);
}
```

g. Window Drawing Procedure

```
// *-------------------------------------*
// | SOURCE FILE: DRWBOX.C               |
// *-------------------------------------*
// | Routine to draw box on the screen.  |
// *-------------------------------------*
#include <string.h>
#include <conio.h>
#include 'rialib.h'
// *-------------------------------------------------*
// | PROCEDURE: draw_box                             |
// *-------------------------------------------------*
// | PURPOSE: Draws a box on the screen.             |
// | ON CALL: ul_X - Upper left corner X coordinate. |
// |          ul_Y - Upper left corner Y coordinate. |
// |          lr_X - Lower right corner X coordinate.|
// |          lr_Y - Lower right corner Y coordinate.|
// | RETURNS: Always returns zero.                   |
// *-------------------------------------------------*
int
draw_box( int ul_X, int ul_Y, int lr_X, int lr_Y ) {
int width, height,x,y,i;
        char Line[81];

        _wscroll = 0;
        width  = lr_X - ul_X - 1;
        height = lr_Y - ul_Y - 1;
        memset( Line, UPPERLEFT, 1 );
        memset( Line+1, HORIZ_SIDE, width );
        memset( Line+1+width, UPPERRIGHT, 1 );
        memset( Line+2+width, 0, 1 );
        x = wherex();
        y = wherey();
        gotoxy( ul_X, ul_Y );
        cputs( Line );
        memset( Line, VERT_SIDE, 1 );
        memset( Line+1, BLANK , width );
        memset( Line+1+width, VERT_SIDE, 1 );
        for ( i = 1; i <= height; i++ ) {
                gotoxy( ul_X, ul_Y + i );
                cputs( Line );
        }
        memset( Line, LOWERLEFT, 1 );
        memset( Line+1, HORIZ_SIDE, width );
        memset( Line+1+width, LOWERRIGHT, 1 );
        gotoxy( ul_X, ul_Y + height + 1 );
        cputs( Line );
        gotoxy ( x, y );
        _wscroll = 1;
        return (0);
}
```

h. File Opening Procedure

```
// *-------------------------------------*
// | SOURCE FILE: GETFILE.C              |
// *-------------------------------------*
// | Routine to open next file on floppy.|
// *-------------------------------------*
#include <string.h>
#include <alloc.h>
#include <dos.h>
#include <dir.h>
#include <stdio.h>
#include 'rialib.h'
// *-------------------------------------------------*
// | PROCEDURE: Get_File                             |
// *-------------------------------------------------*
// | PURPOSE:   Opens the next file on the floppy drive. |
// | ON CALL:   Filename - Buffer for filename.          |
// |            Drive    - Drive to read from.           |
// |            Start    - Flag for findfirst/findnext   |
// | RETURNS:   Pointer to open file structure           |
// |            Returns NULL when no more files.        |
// *-------------------------------------------------*
FILE *
Get_File( char *Filename, char *Drive, int *start ) {
        FILE *Opened_File;
        char File[20];
        int done;
        static struct ffblk ffblk;
        char Error_Msg[100];

        strcpy( File, Drive );
        strcat( File, "*.*" );
        if ( *start ) {
                done = findfirst( File, &ffblk, 0 );
                *start = 0;
            }
          else
              done = findnext( &ffblk );
        if ( done )
                return( NULL );
        strcpy( File, Drive );
        strcat( File, ffblk.ff_name );
        if (( Opened_File = fopen( File, 'r' )) == NULL ) {
                sprintf(Error_Msg,'Error: Can\'t open %s\n', File );
```

```
            die( Error_Msg );
        }
        strcpy( Filename, ffblk.ff_name );
        return ( Opened_File );
}
```

i. Get User Name Procedure

```
// *-------------------------------------------------*
// | SOURCE FILE: GETUNAM.C                          |
// *-------------------------------------------------*
// | Routine to get user name.                       |
// *-------------------------------------------------*
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include 'rialib.h'
// *-------------------------------------------------------------------*
// | PROCEDURE: Get_User_Name                                          |
// *-------------------------------------------------------------------*
// | PURPOSE:    Have the user input their name for use in file        |
// |                  descriptions.                                    |
// | ON CALL:    Name_Window - Description for input name              |
// |                  window.                                          |
// |             User_Name - Points to buffer for user name.           |
// |             Time_Stamp  - File name for time stamp file.          |
// | RETURNS:    Nothing                                               |
// *-------------------------------------------------------------------*

void
Get_User_Name( struct Window_Style *Name_Window,
               char *User_Name, char *Time_Stamp ) {
        char *savebuffer;
        FILE *Time;
        struct text_info ti;

        gettextinfo( &ti );
        savebuffer = Window_Save( Name_Window );
        Window_Box( Name_Window );
        gotoxy( Name_Window->x1+1, Name_Window->y1+1 );
        textcolor( Name_Window->textcolor );
        textbackground( Name_Window->backcolor );
        while (strlen(User_Name) == 0 ) {
                Input_String( User_Name, 12, 0, 0 );
                if (strlen(User_Name) == 0 )
                        Beep();
        }
        if ( (Time=fopen(Time_Stamp, 'w' )) == NULL )
                die( 'Can\'t open timestamp file.\n');
        fprintf( Time, '%s', User_Name );
        fclose( Time );
        Window_Restore( Name_Window, savebuffer );
        textattr( ti.attribute );
}
```

j. High Level Hardware Error Handling Procedure

```
// *-------------------------------------------------*
// | SOURCE FILE: HARDERR.C                          |
// *-------------------------------------------------*
// | High level routine to process hardware errors.  |
// *-------------------------------------------------*
#include <dos.h>
#include <string.h>
#include <stdio.h>
#include 'rialib.h'

static char *error_message[] = {
        'write protect',
        'unknown unit',
        'drive not ready',
        'unknown command',
        'data CRC error',
        'bad request',
        'seek error',
        'unknown media type',
        'sector not found',
        'printer out of paper',
        'write fault',
        'read fault',
        'general failure',
        'reserved',
        'reserved',
        'invalid disk change'
};

// *-------------------------------------------------------------------*
// | PROCEDURE: Hardware_Error                                         |
// *-------------------------------------------------------------------*
// | PURPOSE:    Process hardware errors that occur.                   |
// | ON CALL:    Called by system -                                    |
// |             deverr - Device error code                            |
// |             errval - Error code                                   |
// |             deviceHeader - Pointer to device driver header.       |
// | RETURNS:    Nothing                                               |
// *-------------------------------------------------------------------*
#pragma warn -par
void far Hardware_Error( unsigned deverr, unsigned errval,
                         unsigned far *deviceHeader ) {
        static char message[80];
        int    drive, errorno, request;

        drive = deverr  & 0x00FF;
        errorno = errval & 0x00FF;
        if ( deverr & 0x8000) {
                sprintf(message,'Non-disk related device error.(%s)',
                                error_message[errorno]);
                request = Display_Critical_Error( message );
                _hardretn( request );
        }
        sprintf(message, 'When trying drive %c:, %s error
        occurred.',
                        'A'+ drive, error_message[errorno] );
        _hardresume( Display_Critical_Error( message ) );
}
#pragma warn +par
```

k. String Input Procedure

```
// *-------------------------------------------------*
// | SOURCE FILE: INPUTSTR.C                         |
// *-------------------------------------------------*
// | Get String Input from the user.                 |
// *-------------------------------------------------*
#include <conio.h>
#include <string.h>
#include <process.h>
#include 'rialib.h'

// *-------------------------------------------------*
// | PROCEDURE: Input_String                         |
// *-------------------------------------------------*
// | PURPOSE: Get a string from the user.            |
// | ON CALL: string - buffer to store string.       |
// |          width - Size of string buffer          |
// |          position - Beginning position in buffer |
```

```
// |          describe - Enables F2 key if TRUE          |
// |  RETURNS: INPUT_OK if string was entered.            |
// |           INPUT_BAD if string input was aborted.     |
// *------------------------------------------------------*
int
Input_String( char *string, int width, int position, int describe ) {
      int x,y, xhome;
      int done=0;
      int i = 0,j;
      char blank[80], ch;

      i = position;
      xhome = x = wherex();
      xhome = xhome - position;
      y = wherey();

      while ( !done ) {
      ch = getch();
      if ( ch == 0 ) {
                  ch = getch();
                  switch (ch) {
                        case F2:
                              if (!describe) {
                                    Beep();
                                    break;
                              }
                              return ( INPUT_BAD );
                        case RIGHT:
                              if (i >= strlen(string) ) {
                                    Beep();
                                    break;
                              }
                              ++x;
                              ++i;
                              break;
                        case LEFT:
                              if ( i <= 0 ) {
                                    Beep();
                                    break;
                              }
                              --x;
                              --i;
                              break;
                        case DELETE:
                              if (string[i] == 0 ) {
                              Beep();
                              break;
                              }
                              strdel( string, i );
                              break;
                        case HOME:
                              x = xhome;
                              i = 0;
                              break;
                        case END:
                              x = xhome + ( strlen( string ) <
                                    width ? strlen(string) :
                                    strlen(string - 1)) ;
                              i = x - xhome;
                              break;
                        default:
                              Beep();
                              break;
                  }
      }
      else
      {
            switch (ch) {
                  case RETURN:
                        done = 1;
                        break;
                  case CNTRL_C:
                        exit(1);
                  case BACKSPACE:
                        if (i==0 ) {
                              Beep();
                              break;
                        }
                        --i;
                        --x;
                        strdel( string, i );
                        break;
                  case ESC:
                        i = -1;
                        done = 1;
                        break;
                  default:
                        if (i >= width  || ch < 32 ||
                              strlen(string) >= width) {
                              Beep();
                              break;
                        }
                        strins(string, ch, i, width);
                        ++i;
                        ++x;
                        break;
                  }
            }
      gotoxy(xhome,y);
      strcpy(blank,'');
      for (j = 0; j<width-1; j++ )
            strcat(blank,' ');
      cputs(blank);
      gotoxy(xhome,y);
      cputs(string);
      gotoxy(x,y);
      }
      if (i < 0 )
      return( INPUT_BAD );
      return ( INPUT_OK );
}
```

I. Prompt to Overwrite File Procedure

```
// *------------------------------------------------------*
// | SOURCE FILE: OVERWRIT.C                              |
// *------------------------------------------------------*
// | Routine to ask permission to overwrite files.        |
// *------------------------------------------------------*
#include <conio.h>
#include <ctype.h>
#include 'rialib.h'

struct Window_Style Overwrite_Window  = {WHITE, CYAN,
                  YELLOW, 2, 10,79, 13, WHITE, CYAN, " };

#define NOT_DETERMINED 5

// *------------------------------------------------------*
// | PROCEDURE: Overwrite                                 |
// *------------------------------------------------------*
// | PURPOSE: Asks user if it is OK to overwrite a file.  |
// | ON CALL: Filename - Name of file to overwrite.       |
// | RETURNS:   TRUE if OK to overwrite.                  |
// |            FALSE if not OK to overwrite.             |
// *------------------------------------------------------*
int
```

```
Overwrite( char *Filename ) {
        char *savebuffer, ch;
        struct text_info ti;
        int overwrite = NOT_DETERMINED;
        char msg[80];

        gettextinfo( &ti );
        savebuffer = Window_Save( &Overwrite_Window );
        Window_Box( &Overwrite_Window );
        gotoxy( Overwrite_Window.x1+2, Overwrite_Window.y1+1
        );
        textcolor( Overwrite_Window.textcolor );
        textbackground( Overwrite_Window.backcolor );
        sprintf(msg,'File %s already exists.',Filename );
        Center_Text( msg, Overwrite_Window );
        gotoxy( Overwrite_Window.x1+2, Overwrite_Window.y1+2
        );
        Center_Text('Do you want to overwrite it (y/n)?',
                Overwrite_Window);
        while ( overwrite == NOT_DETERMINED ) {
                ch = getch();
                tolower(ch);
                switch (ch) {
                        case 'y':
                                overwrite = TRUE;
                                break;
                        case 'n':
                                overwrite = FALSE;
                                break;
                        default:
                                Beep();
                                break;
                }
        }
        Window_Restore( &Overwrite_Window, savebuffer );
        textattr( ti.attribute );
        return( overwrite );
}
```

m. Prompt for New Disk Procedure

```
// *-------------------------------------------*
// | SOURCE FILE: PNEWDISK.C                   |
// *-------------------------------------------*
// | Prompts the user to insert a new disk.    |
// *-------------------------------------------*
#include <dos.h>
#include <dir.h>
#include <string.h>
#include <stdio.h>
#include <bios.h>
#include 'rialib.h'

#define BOX_UL_X 6
#define BOX_UL_Y 11
#define BOX_LR_X 73
#define BOX_LR_Y 13


// *-------------------------------------------*
// | PROCEDURE: Prompt_New_Disk                |
// *-------------------------------------------*
// | PURPOSE: Prompt the user to insert a new floppy disk. |
// | ON CALL: Prompt_Disk - Flag to indicate if Prompting. |
// |          Drive - Disk Drive in use.       |
// | RETURNS: Nothing                          |
// *-------------------------------------------*
void
Prompt_New_Disk( char *Prompt_Disk, char *Drive ) {
struct ffblk File_Block;
struct text_info ti;
int done;
FILE *dummy;
char box_save[( BOX_LR_X-BOX_UL_X + 2) * (BOX_LR_Y
        - BOX_UL_Y +2 ) * 2 ];
char Search_String[20];
char titleLeft[2]={LEFTCROSS,0},
        titleRight[2]={RIGHTCROSS,0};

*Prompt_Disk = 1;
gettextinfo( &ti );
gettext( BOX_UL_X, BOX_UL_Y, BOX_LR_X, BOX_LR_Y,
        box_save );
textattr( RED*16+WHITE );
draw_box( BOX_UL_X, BOX_UL_Y, BOX_LR_X,
        BOX_LR_Y );
gotoxy( BOX_UL_X+3, BOX_UL_Y+1 );
gotoxy( BOX_UL_X+3, BOX_LR_Y );
cputs( titleRight );
textattr( RED*16+YELLOW );
cputs('Press the ENTER key when ready');
textattr( RED*16+WHITE );
cputs( titleLeft );
_setcursortype(_NOCURSOR);
gotoxy( BOX_UL_X+3, BOX_UL_Y+1 );
cputs('Please place the RIA data disk into the 5¬\'
        EXTERNAL drive.' );
while ( getch() != 13 ) {
        sound(110);
        delay(200);
        nosound();
}
puttext( BOX_UL_X, BOX_UL_Y, BOX_LR_X, BOX_LR_Y,
        box_save );
_setcursortype(_NORMALCURSOR);
biosdisk( RESET_DRIVE, 0, 0, 0, 0, 0, NULL);
lowvideo();
strcpy(Search_String, Drive);
strcat( Search_String, 'lscdsk.tst');
/*Loop until we found disk */
while ((dummy=fopen ( Search_String,'w' )) == NULL );
fclose (dummy);
unlink( Search_String );
strcpy(Search_String, Drive);
strcat( Search_String, '*.unk');
done = 0;
while (!done) {
        done = findfirst( Search_String, &File_Block, 0 ) ^ 1;
        if (!done) {
                strcpy(Search_String, Drive);
                strcat( Search_String, '*.rec');
                done = findfirst( Search_String, &File_Block,
                        0 ) ^ 1;
                if (!done) {
                        gettext( BOX_UL_X, BOX_UL_Y,
                                BOX_LR_X, BOX_LR_Y+1,
                                box_save );
                        textattr( RED*16+WHITE );
                        draw_box( BOX_UL_X, BOX_UL_Y,
                                BOX_LR_X, BOX_LR_Y+1 );
                        gotoxy( BOX_UL_X+3, BOX_UL_Y+1 );
                        _setcursortype(_NOCURSOR);
                        cputs( '      This disk does not contain
                                RIA data files.   ');
                        gotoxy( BOX_UL_X+3, BOX_UL_Y + 2 );
                        cputs( ' Please insert a an RIA data disk,
                                then push the ENTER key.' );
```

```
                while ( getch() != 13 ) {
                    sound(110);
                    delay(200);
                    nosound();
                }
                puttext( BOX_UL_X, BOX_UL_Y,
                    BOX_LR_X, BOX_LR_Y+1,
                    box_save );
                _setcursortype(_NORMALCURSOR);
                biosdisk( RESET_DRIVE, 0, 0, 0, 0, 0,
                    NULL);
            }
        }
    }
    *Prompt_Disk = 0;
    textattr( ti.attribute );
}
```

n. Read Descriptions Procedure

```
// *-------------------------------------------*
// | SOURCE FILE: READDESC.C                   |
// *-------------------------------------------*
// | Procedure to load descriptions.           |
// *-------------------------------------------*
#include <dos.h>
#include <dir.h>
#include <alloc.h>
#include <string.h>
#include <stdio.h>
#include 'rialib.h'


// *-------------------------------------------*
// | PROCEDURE: Read_Descriptions              |
// |-------------------------------------------|
// | PURPOSE: Reads file descriptions into memory. |
// | ON CALL: datapath - Path to RIA data files. |
// |          fcount     - Pointer to description count. |
// |          file_desc - Pointer to point to array of descriptions. |
// | RETURNS: Nothing                          |
// *-------------------------------------------*
void
Read_Descriptions( char *datapath, int *fcount, char
        **file_desc ) {
    char Filename[120];
    int notfound,i=0, ch;
    char * in_desc;
    int line_end=0;
    struct ffblk ffblk;
    FILE *Desc_File;

    *fcount = -1;
    strcpy(Filename, datapath );
    strcat(Filename, 'descript.ion' );
    in_desc = allocate( 4096 );
    notfound = findfirst( Filename, &ffblk, FA_HIDDEN );
    if ( !notfound ) {
        if ((Desc_File = fopen( Filename, 'rb' )) == NULL)
            die('Couldn\'t open description file.\n');
        while(( ch = fgetc( Desc_File ) ) != EOF ) {
            switch( ch ) {
                case CR:
                case LF:
                case 26:
                if ( !line_end ) {
                    line_end = 1;
                    in_desc[i] = 0;
                    file_desc[++(*fcount)] = allocate(
                        strlen(in_desc) + 1 );
                    strcpy( file_desc[(*fcount)],
                        in_desc );
                    i = 0;
                }
                break;
                default:
                if ( i <= 4096 ) {
                    in_desc[i] = ch;
                    ++i;
                }
                line_end = 0;
                break;
            }
            if ( ch == 26 ) break;
        }
        if (ch == EOF && !line_end ) {
            in_desc[i] = 0;
            file_desc[++(*fcount)] = allocate( strlen(in_desc)
                + 1 );
            strcpy( file_desc[(*fcount)], in_desc );
        }
        fclose ( Desc_File );
    }
    (*fcount)++;
    free (in_desc );
}
```

o. Description Saving Procedure

```
// *-------------------------------------------*
// | SOURCE FILE: SAVEDESC.C                   |
// |-------------------------------------------|
// | Procedure to save new descriptions        |
// *-------------------------------------------*
#include <string.h>
#include <stdio.h>
#include <dos.h>
#include <alloc.h>
#include <conio.h>
#include 'rialib.h'

// *-------------------------------------------*
// | PROCEDURE: Save_Desc                      |
// |-------------------------------------------|
// | PURPOSE: Save the newly created descriptions. |
// | ON CALL: datapath - Path to where data files are located. |
// |          file_desc - Pointer to array of old descriptions |
// |          new_desc - Pointer to array of new descripts. |
// |          Copied_Files - Number of files copied (# new |
// |                descriptions)              |
// |          fcount - Number of old descriptions. |
// | RETURNS: Nothing                          |
// |-------------------------------------------|
void
Save_Desc( char *datapath, char **file_desc, char **new_desc,
    int Copied_Files, int fcount ) {
    char *tmp_desc;
    int i, j, len;
    char *extra;
    FILE *Desc_File;

    tmp_desc = allocate ( 4096 );
    strcpy( tmp_desc, datapath );
    strcat( tmp_desc, 'descript.ion' );
    if ((Desc_File = fopen( tmp_desc, 'w' )) == NULL)
        die('Couldn\'t open description file.\n');
    for ( i = 0; i< Copied_Files; i++ ) {
```

```
            extra = new_desc[i];
            for (j=0; j<strlen(new_desc[i]); j++ ) {
                  if ( extra[j] == ' ' ) {
                        len = j;
                        j = 5000;
                  }
            }
            strcpy( tmp_desc, new_desc[i] );
            for ( j=0; j < fcount; j++ ) {
                  if (strncmpi(new_desc[i], file_desc[j], len ) == 0 )
                  {
                        extra = strchr( file_desc[j], 4 );
                        if (extra != NULL )
                              strcat( tmp_desc, extra );
                        j = 5000;
                        file_desc[j] = NULL;
                  }
            }
            fprintf( Desc_File,"%s\n", tmp_desc );
      }
      for (i = 0; i< fcount+1; i++ ) {
            if ( file_desc[i] != NULL )
                  fprintf( Desc_File,"%s\n", file_desc[i] );
      }
      fclose( Desc_File );
      strcpy( tmp_desc, datapath );
      strcat( tmp_desc, "descript.ion");
      _dos_setfileattr( tmp_desc, _A_HIDDEN );
      free ( tmp_desc );
}
```

p. Screen Set-up Procedure

```
// *-------------------------------------------------*
// | SOURCE FILE: SCRNSU.C                           |
// |-------------------------------------------------|
// | Draws screen windows.                           |
// *-------------------------------------------------*
#include <stdio.h>
#include "rialib.h"

// *-------------------------------------------------*
// | PROCEDURE: Screen_Setup                         |
// |-------------------------------------------------|
// | PURPOSE: Draw a series of Windows on the screen.|
// | ON CALL: windowList - Pointer to a list of window|
// |                       definition structs.       |
// | RETURNS: Nothing                                |
// *-------------------------------------------------*
void
Screen_Setup( struct Window_Style **windowList ) {
      while ( *windowList != NULL ) {
            Window_Box( *windowList );
            windowList++;
      }
}
```

q. Character Deletion Procedure

```
// *-------------------------------------------------*
// | SOURCE FILE: STRDEL.C                           |
// |-------------------------------------------------|
// | Procedure to delete a character from string.    |
// *-------------------------------------------------*
#include <string.h>

// *-------------------------------------------------*
// | PROCEDURE: strdel                               |
// |-------------------------------------------------|
// | PURPOSE: Deletes a character from the middle of a string. |
// | ON CALL: string - Pointer to string Buffer      |
// |          posit  - Position of character to delete.|
// | RETURNS: New position for cursor                |
// |          -1 if can't do anything                |
// *-------------------------------------------------*
int
strdel( char *string, int posit ) {
      int length, i;

      length = strlen(string);
      if ( posit >= length || length == 0 )
            /* The string is empty, or we are beyond end of string.
               */
            return (-1);
      if ( posit == length-1 ) { /* End of the string */
            string[posit]=0;
            return (posit);
      }
      for ( i = posit; i< length; i++ )
            string[i] = string[i+1];
      return ( posit );
}
```

r. Character Insertion Procedure

```
// *-------------------------------------------------*
// | SOURCE FILE: STRINS.C                           |
// |-------------------------------------------------|
// | Procedure to insert character in a string.      |
// *-------------------------------------------------*
#include <string.h>

// *-------------------------------------------------*
// | PROCEDURE: strins                               |
// |-------------------------------------------------|
// | PURPOSE: Inserts a character within a string.   |
// | ON CALL: string - Pointer to string buffer.     |
// |          ch     - Character to insert           |
// |          posit  - Position to insert character. |
// |          size   - Length of string.             |
// | RETURNS: Next position for cursor.              |
// *-------------------------------------------------*
int
strins( char *string, char ch, int posit, int size ) {
      int length, i;

      length = strlen(string);
      if ( posit > size - 1 || posit > length )
            /* The string is already full, so we can't add any more
               */
            return (-1);
      if ( posit == length ) { /* End of the string */
            string[posit]=ch;
            string[posit+1]=0;
            return (posit+1);
      }
      for ( i = length; i>= posit; i- )
            string[i+1] = string[i];
      string[posit] = ch;
      return ( posit + 1 );
}
```

s. String Storage Procedure

```
// *-------------------------------------------------*
// | SOURCE FILE: STRSTORE.C                         |
// |-------------------------------------------------|
// | Routine to store a string in the far Heap       |
```

```
// *-------------------------------------------*
#include <string.h>
#include <alloc.h>
#include 'rialib.h'

// *-------------------------------------------*
// | PROCEDURE: strstore                       |
// |-------------------------------------------|
// | PURPOSE: Allocates space for a string on the heap |
// |          and copies it.                   |
// | ON CALL: string - pointer to the string to store |
// | RETURNS: pointer to stored string.        |
// *-------------------------------------------*
char *
strstore ( char *string ) {
      char *temp;

      temp = allocate( strlen( string) + 1 );
      strcpy( temp, string );
      return ( temp );
}
```

t. Window Drawing Procedure

```
// *-------------------------------*
// | SOURCE FILE: WINBOX.C         |
// |-------------------------------|
// | Procedure to draw windows.    |
// *-------------------------------*
#include <conio.h>
#include <string.h>
#include 'rialib.h'

// *-------------------------------------------*
// | PROCEDURE: Window_Box                     |
// |-------------------------------------------|
// | PURPOSE: Draw a window on the screen      |
// | ON CALL: Window - Pointer to a window definition |
// ??|          structure.                     |
// | RETURNS: Nothing                          |
// *-------------------------------------------*
void
Window_Box( struct Window_Style *Window ) {
      struct text_info ti;
      int offset;
      char titleLeft[2]={LEFTCROSS,0},
           titleRight[2]={RIGHTCROSS,0};

      gettextinfo( &ti );
      textcolor( Window->forecolor );
      textbackground( Window->backcolor );
      draw_box( Window->x1, Window->y1, Window->x2,
           Window->y2 );
      offset = (( Window->x2 - Window->x1 + 1 ) - strlen( Window-
           >title ) - 2 ) / 2;
      if (offset < 0 ) offset = 0;
      if ( strlen(Window->title) != 0 ) {
           gotoxy( Window->x1+offset, Window->y1 );
           cputs( titleRight );
           textcolor( Window->titlefore );
           textbackground( Window->titleback );
           cputs(Window->title);
           textcolor( Window->forecolor );
           textbackground( Window->backcolor );
           cputs( titleLeft );
      }
      textattr( ti.attribute );
}
```

u. Screen Background Restoring Procedure

```
// *-------------------------------*
// | SOURCE FILE: WINREST.C        |
// |-------------------------------|
// | Restore a saved portion of the screen. |
// *-------------------------------*
#include <conio.h>
#include <alloc.h>
#include 'rialib.h'

// *-------------------------------------------*
// | PROCEDURE: Window_Restore                 |
// |-------------------------------------------|
// | PURPOSE: Restores a saved portion of the screen. |
// | ON CALL: Window - Pointer to window definition |
// |          for the window we are about     |
// |          to close.                        |
// |          savebuffer - Pointer to saved data. |
// | RETURNS: Nothing                          |
// *-------------------------------------------*
void
Window_Restore( struct Window_Style *Window, char
           *savebuffer ) {

      puttext( Window->x1, Window->y1, Window->x2, Window-
           >y2, savebuffer );
      free (savebuffer);
}
```

v. Screen Background Saving Procedure

```
// *-------------------------------*
// | SOURCE FILE: WINSAV.C         |
// |-------------------------------|
// | Procedure to save a portion of the screen. |
// *-------------------------------*
#include <alloc.h>
#include <conio.h>
#include <stdio.h>
#include <process.h>
#include 'rialib.h'

// *-------------------------------------------*
// | PROCEDURE: Window_Save                    |
// *-------------------------------------------*
// | PURPOSE: To save a portion of the screen for restore |
// |          later.                           |
// | ON CALL: Window - Pointer to window definition for |
// |          window we are about to open      |
// | RETURNS: Pointer to the buffer with the saved data. |
// *-------------------------------------------*
char *
Window_Save( struct Window_Style *Window ) {
      int buffersize;
      char *savebuffer;
      buffersize = ( Window->x2 - Window->x1 + 1 ) * (Window-
           >y2 - Window->y1 + 1 ) * 2;
      if (( savebuffer = (char *) malloc( buffersize ) )== NULL ) {
           clrscr();
           fprintf(stderr, 'Error allocating temporary buffer.\n');
           exit (1);
      }
      gettext( Window->x1, Window->y1, Window->x2, Window-
           >y2, savebuffer );
      return ( savebuffer );
```

}

w. Critical Error Handling Procedure

```
                        TITLE  CRITICAL.ASM  Critical Error Handler
                        PAGE  55,132
                        ;*-------------------------------------------------*
                        ;| Critical Error routines for use with RIA Program. |
                        ;*-------------------------------------------------*
                        .MODEL small

BUFFER_SIZE             EQU 10240
UPPERLEFT               EQU 0DAH
UPPERRIGHT              EQU 0BFH
LOWERLEFT               EQU 0C0H
LOWERRIGHT              EQU 0D9H
VERT_SIDE               EQU 0B3H
HORIZ_SIDE              EQU 0C4H
SPACE                   EQU 020H
GET_VIDEO_MODE          EQU 0FH
GET_CURSOR              EQU 03H
GET_KEY_STATUS          EQU 1
GET_KEY                 EQU 0
IGNORE                  EQU 0
RETRY                   EQU 1
ABORT                   EQU 2
FAIL                    EQU 3

                        .DATA
                        EXTRN   _Prompt_Disk:BYTE
Init_Msg                DB      'Please place the RIA data disk into the 5-¼" EXTERNAL Drive.'
Len_Init_Msg            EQU     $ - Init_Msg
Init_Msg_2              DB      'Press any key to continue.'
Len_Init_Msg_2          EQU     $ - Init_Msg_2
Abort_Msg               DB      'Correct error then press a key: A)bort, R)etry, I)gnore, or F)ail.'
Len_Abort_Msg           EQU     $ - Abort_Msg
Video_Segment           DW      0
Vid_Offset              DW      0
Cursor_Type             DW      0
Cursor_Position         DW      0
Active_Page             DB      0
Save_Buffer             DB      64 DUP ('SCREENBUFF')

                        .CODE
                        ASSUME CS:@code, DS:@data

;*-------------------------------------------------*
;| PROCEDURE: Sound                                |
;*-------------------------------------------------*
;| PURPOSE:   Sound the speaker for a specific duration and frequency.
;| ON CALL:   BX has the duration in msec          |
;|            DI has the frequency in Hertz        |
;| RETURNS:   Nothing                              |
;*-------------------------------------------------*
Sound   PROC    NEAR
        ;*-------------------------------------------------*
        ;| Actual duration is in multiples of 55 msec as we are using
        ;| timer ticks.                                    |
        ;*-------------------------------------------------*
        PUSH    AX              ; Save registers.
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    DI

        MOV     AL,0B6H         ; Load Timer notification code into AL.
        OUT     43H,AL          ; Notify Timer that count is coming.
        ;*-------------------------------------------------*
```

```
        ; | Timer frequency is 1.19318 MHz ( 144F38H )                    |
        ; | Timer count is (Timer Frequency) / (Sound Frequency)          |
        ; *---------------------------------------------------------------*
        MOV     DX,14H          ; Load DX:AX with Timer frequency
        MOV     AX,4F38H        ;
        DIV     DI              ; Divide by sound frequency.
        OUT     42H,AL          ; Send low order byte of count to timer.
        MOV     AL,AH           ; Move high order byte of count into AL.
        OUT     42H,AL          ; Send high order byte of count to timer.
        IN      AL,61H          ; Read byte from Programmable Peripheral Interface Chip (PPI).
        MOV     AH,AL           ; Save it in AH.
        OR      AL,3            ; Set Bits 1 and 2.
        OUT     61H,AL          ; Tell PPI to turn the speaker on.
        PUSH    AX              ; Save AX
        MOV     AX, BX          ; Get the number of msec in AX.
        MOV     CX, 55          ; Load CX with the number of msec/tick.
        DIV     CL              ; Get number of timer ticks into AL.
        CMP     AH, 27          ; Do we need to round up (remainder>27)?
        JLE     Sound_1   ;     If no, jump.
        INC     AL              ; Round up.
Sound_1:
        XOR     AH,AH           ; Clear high order byte.
        MOV     BX,AX           ; Save number of ticks.
        CALL    Get_Ticks       ; Get current ticks
        MOV     DI, AX
        MOV     CX, DX          ; CX:DI has starting timer count.
Sound_2:
        CALL    Get_Ticks       ; Get the current Tick Count.
        SUB     AX, DI          ; Determine Ticks elapsed.
        SBB     DX, CX
        CMP     AX, BX          ; Has the time expired?
        JLE     Sound_2         ; If no, loop until done.

        POP     AX              ; Restore PPI port value in AH.
        MOV     AL,AH           ; Move original PPI port into AL.
        OUT     61H,AL          ; Tell PPI to turn off the speaker.

        POP     DI              ; Restore Registers.
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET

Sound   ENDP


; *---------------------------------------------------------------*
; | PROCEDURE: Get_Ticks                                          |
; *---------------------------------------------------------------*
; | PURPOSE:   Gets the current timer tick count in DX:AX         |
; | ON CALL:   Nothing                                            |
; | RETURNS:   DX:AX has the timer tick count.                    |
; *---------------------------------------------------------------*
Get_Ticks PROC    NEAR
        PUSH    DI
        PUSH    DS
        PUSH    ES
        MOV     AX, ROMBIOS     ; Load the ROM BIOS segment.
        MOV     ES, AX
        MOV     DI, 06CH        ; Point to the timer ticks.
        MOV     AX, ES:DI       ; Get timer ticks into DS:AX
        MOV     DX, ES:DI[2]    ; Timer Count now in DX:AX
        POP     ES
        POP     DS
        POP     DI
        RET
Get_Ticks ENDP
```

```
// *-------------------------------------------------------------*
// | PROCEDURE: _Display_Critical_Error                          |
// *-------------------------------------------------------------*
// | PURPOSE: Display critical error message, or prompt for new disk if |
// |          the drive was empty when we tried to read the RIA data.   |
// | ON CALL: Error_Message - Points to the NULL terminated message.    |
// |          _Prompt_Disk  - Global flag set in the calling program.   |
// |                          Determines if we prompt for a new disk.   |
// | RETURNS: Returns control to hardware error handler.         |
// *-------------------------------------------------------------*
                    PUBLIC  _Display_Critical_Error
_Display_Critical_Error PROC    NEAR
                    ARG     Error_Message:WORD

                    PUSH    BP
                    MOV     BP,SP

                    PUSH    BX
                    PUSH    CX
                    PUSH    DX
                    PUSH    DI
                    PUSH    SI
                    PUSH    ES
                    MOV     AH,GET_VIDEO_MODE       ; Load BIOS function 'Get Video Mode'
                    INT     10H                     ; Get the video mode.
                    MOV     Active_Page, BH         ; Save the active video page.
                    MOV     Video_Segment, 0B800H   ; Assume for now that this is not a mono card.
                    CMP     AL,07H                  ; Are we on a mono card?
                    JNE     Display_Critical_Error_10 ;         If no, jump.
                    MOV     Video_Segment, 0B000H   ; Set video segment for monochrome adapter.
Display_Critical_Error_10:
                    MOV     AL,AH                   ; Load the page width into AL.
                    XOR     AH,AH                   ; Zero out the high order byte.
                    PUSH    BX                      ; Save the active display page.
                    MUL     BH                      ; Multipy by the page number.
                    MOV     BH,25                   ; Load the number of lines per page.
                    MUL     BH                      ; Multiply by the number of lines.
                    MOV     Vid_Offset, AX          ; Save the Video offset.
                    MOV     AH, GET_CURSOR          ; Load BIOS function 'Get Cursor Position'
                    POP     BX                      ; Get active page in BX.
                    INT     10H                     ; Get the cursor position.
                    MOV     Cursor_Type, CX         ; Save the cursor type.
                    MOV     Cursor_Position, DX     ; Save the cursor position.
                    PUSH    DS                      ; Make our data segment the extra segment.
                    POP     ES
                    CMP     _Prompt_Disk,0          ; Are we prompting for a new disk?
                    JE      Display_Critical_Error_20 ;         If no, jump.
                    MOV     AX,Len_Init_Msg         ; Load the length of the disk prompt.
                    MOV     DI, Len_Init_Msg+4      ; Load size of message with 2 spaces at each end.
                    JMP     Display_Critical_Error_50 ; Jump over other error processing stuff.
Display_Critical_Error_20:
                    MOV     DI,[Error_Message]      ; Load offset to error message.
                    PUSH    DI                      ; Save it on the stack.
                    MOV     CX,100                  ; Load maximum line length.
                    MOV     AL,0                    ; Load a zero to search for.
                    REPNE   SCASB                   ; Search for 0 termination of message.
                    CMP     CX,0                    ; Was the string terminated?
                    JNE     Display_Critical_Error_30 ;         If yes, jump.
                    POP     AX                      ; Clear the offset from the stack.
                    MOV     AX,-1                   ; Signal error.
                    JMP     Display_Critical_Error_190 ; Error no end of string found, Exit.
Display_Critical_Error_30:
                    DEC     DI                      ; Points to zero at end of line.
                    POP     AX                      ; Get the starting point of the string.
                    SUB     DI,AX                   ; DI has string length.
                    MOV     AX,DI                   ; Save the message length in DI.
```

```
                CMP     DI, Len_Abort_Msg       ; Is it longer than the abort message?
                JG      Display_Critical_Error_40 ;         if yes then continue.
                MOV     DI, Len_Abort_Msg       ; Set box width to the size of the abort message.
Display_Critical_Error_40:
                ADD     DI,4                    ; Add one space, and the edges.
                CMP     DI,80                   ; Are we larger than 80.
                JLE     Display_Critical_Error_50 ;         if no, then jump.
                MOV     DI,80                   ; Set box width to the screen width.
Display_Critical_Error_50:
                MOV     DX,DI                   ; Save the box width in DX.
                MOV     DH,AL                   ; Store the error message length in AL.
                MOV     AX,80                   ; Load AX with the screen width.
                SUB     AX,DI                   ; Determine how much space is left.
                AND     AX,11111110B            ; Make it even.
                ADD     AX, 10*80*2             ; Get Offset.
                ADD     AX,Vid_Offset           ; Add page offset.
                MOV     BX,AX                   ; Store the offset in AX
                ;*-------------------*
                ;| Store the screen  |
                ;*-------------------*
                PUSH    DS                      ; Make our data segment our extra segment.
                POP     ES
                PUSH    DS                      ; Save out data segment
                MOV     AX,Video_Segment        ; Make the video segment our Data Segment.
                MOV     DS,AX                   ;
                MOV     SI, BX                  ; Point to the upper left box corner.
                MOV     DI, OFFSET Save_Buffer  ; Point to our storage buffer.
                MOV     CX,4                    ; Load number of lines to save.
Display_Critical_Error_60:
                PUSH    CX                      ; Save line count.
                MOV     CL,DL                   ; Load Box width into CL.
                XOR     CH,CH                   ; Clear the high order byte.
                PUSH    SI
                REP     MOVSW                   ; Save the line in the buffer.
                POP     SI
                ADD     SI,160                  ; Point to the next line.
                POP     CX                      ; Restore the line count.
                LOOP    Display_Critical_Error_60 ; Continue until done.
                POP     DS                      ; Restore the data segment.
                MOV     AX,Video_Segment        ; Load the video segment into AX.
                MOV     ES,AX                   ; Make the video segment our extra segment.
                MOV     DI,BX                   ; Make DI point to the upper left corner of the box.
                MOV     AH, BYTE PTR ES:DI[1]   ; Get the attribute there.
                MOV     AL,AH                   ; Make it reverse video.
                AND     AL,01110000B            ; Get the background attribute.
                AND     AH,00000111B            ; Get the foreground attribute
                MOV     CL,4                    ; Get ready to swap attributes.
                SHL     AH,CL                   ; Make the foreground into the background.
                SHR     AL,CL                   ; Make the background into the foreground.
                OR      AH,AL                   ; Put the attributes together.
                OR      AH,00001000B            ; Turn on High intensity
                OR      AH,10000000B            ; Turn on Blink.
                MOV     AL,UPPERLEFT            ; Load character for upper left corner.
                STOSW                           ; Put it on the screen.
                MOV     AL,HORIZ_SIDE           ; Load the character for the box top.
                MOV     CX,DX                   ; Load the Box Width.
                SUB     CL,2                    ; Determine how many characters across the top.
                XOR     CH,CH                   ; Clear high order byte.
                PUSH    CX                      ; Save the line fill count.
                REP     STOSW                   ; Make the top of the box.
                POP     CX                      ; Restore the line fill count.
                MOV     AL,UPPERRIGHT           ; Load the upper right corner character.
                STOSW                           ; Store the upper right corner in the video buffer.
                MOV     DI,BX                   ; Load the upper left corner into DI.
                ADD     DI,160                  ; Point to the second line.
                PUSH    DI                      ; Save the offset to line 2.
                MOV     AL,VERT_SIDE            ; Load the vertical edge character.
                PUSH    AX                      ; Save the left border character.
```

```
                    STOSW                                   ; Put left border on box.
                    MOV     AL,SPACE                        ; Load a blank.
                    PUSH    CX                              ; Save the width count.
                    REP     STOSW                           ; Make the blank part of the box line.
                    POP     CX                              ; Restore the width count.
                    POP     AX                              ; Restore border character
                    STOSW                                   ; Make the right border.
                    POP     DI                              ; Restore pointer to line 2
                    ADD     DI,160                          ; Point to the third line.
                    PUSH    DI                              ; Save the offset to line 2.
                    MOV     AL,VERT_SIDE                    ; Load the vertical edge character.
                    PUSH    AX                              ; Save the left border character.
                    STOSW                                   ; Put left border on box.
                    MOV     AL,SPACE                        ; Load a blank.
                    PUSH    CX                              ; Save the character repeat count.
                    REP     STOSW                           ; Make the blank part of the box line.
                    POP     CX                              ; Restore the character repeat count.
                    POP     AX                              ; Restore border character
                    STOSW                                   ; Make the right border.
                    POP     DI                              ; Restore pointer to thrid line.
                    ADD     DI,160                          ; Point to the bottom of the box.
                    MOV     AL,LOWERLEFT                    ; Load lower left corner.
                    STOSW                                   ; Store it in the video buffer.
                    MOV     AL,HORIZ_SIDE                   ; Load the box bottom.
                    PUSH    CX                              ; Save the count.
                    REP     STOSW                           ; Put it in the video buffer.
                    POP     CX                              ; Restore the count.
                    MOV     AL,LOWERRIGHT                   ; load the lower right corner.
                    STOSW                                   ; Put it in the video buffer.
                    AND     AH,01111111B                    ; Turn off blink.
                    PUSH    CX                              ; Save the count.
                    SUB     CL,DH                           ; Determine Space remaining.
                    AND     CL,11111110B                    ; Make sure it is even.
                    MOV     DI,BX                           ; Load the upper left corner of the box.
                    ADD     DI,160                          ; Move to the next line.
                    PUSH    DI                              ; Save pointer to next line.
                    XOR     CH,CH                           ; Clear high order byte of counter.
                    ADD     DI,CX                           ; Indent if necessary.
                    MOV     SI,[Error_Message]              ; Point to the Error message.
                    CMP     _Prompt_Disk,0                  ; Do we need to prompt for a data disk?
                    JE      Display_Critical_Error_70       ;       If no, then jump.
                    MOV     SI,OFFSET Init_Msg              ; Load the disk prompt message.
Display_Critical_Error_70:
                    MOV     CL,DH                           ; Load the error message size into CX.
                    XOR     CH,CH                           ; Clear the high order byte of the counter.
Display_Critical_Error_80:
                    LODSB                                   ; Get a character from the message.
                    STOSW                                   ; Display it on the screen.
                    LOOP    Display_Critical_Error_80       ; Continue till message displayed.
                    POP     DI                              ; Restore pointer to beginning of line.
                    POP     CX                              ; Restore the count.
                    ADD     DI,160                          ; Point to beginning of next line.
                    SUB     CL, Len_Init_Msg_2              ; Load length of 'Press Any Key to continue' message.
                    CMP     _Prompt_Disk,0                  ; Do we need to promp for a new disk?
                    JNE     Display_Critical_Error_100      ;       If yes, then jump.
                    SUB     CL, Len_Abort_Msg               ; Load the length of the abort message.
Display_Critical_Error_100:
                    AND     CL,11111110B                    ; Make sure it is even.
                    XOR     CH,CH                           ; Clear high byte of count.
                    ADD     DI,CX                           ; Point to start of line.
                    MOV     CX,Len_Abort_Msg                ; Load the length of the abort message.
                    MOV     SI,OFFSET Abort_Msg             ; Point to the abort message.
                    CMP     _Prompt_Disk,0                  ; Are we prompting for a new disk?
                    JE      Display_Critical_Error_110      ;       If no, then jump.
                    MOV     CX,Len_Init_Msg_2               ; Load length of 'Press any   ' message.
                    MOV     SI,OFFSET Init_Msg_2            ; Point to 'Press any   ' message.
Display_Critical_Error_110:
                    LODSB                                   ; Load a byte of the message.
```

```
                        STOSW                                   ; Display it on the screen.
                        LOOP    Display_Critical_Error_110      ; Loop until done.

Display_Critical_Error_120:
                        PUSH    DI                              ; Save DI.
                        PUSH    BX                              ; Save BX.
                        MOV     DI,300                          ; Set frequency to 300 Hertz.
                        MOV     BX,150                          ; Set Duration to 150 MS.
                        CALL    Sound                           ; Sound a beep.
                        MOV     DI,110                          ; Set frequency to 110 Hertz.
                        MOV     BX,200                          ; Set Duration to 200 MS.
                        CALL    Sound                           ; Sound a beep.
                        POP     BX                              ; Restore BX, and DI
                        POP     DI                              ;
Display_Critical_Error_130:
                        MOV     AH, GET_KEY_STATUS              ; Load BIOS function get keyboard status.
                        INT     16H                             ; See if a key is waiting.
                        JZ      Display_Critical_Error_130      ;       If no, loop until one is there.
                        MOV     AH, GET_KEY                     ; Load BIOS function get key press.
                        INT     16H                             ; Get the key.
                        CMP _   Prompt_Disk, 0                  ; Are we prompting for a new disk?
                        JNE     Display_Critical_Error_170      ;       If yes, jump.
                        ;*-------------------------------------*
                        ;| Processing disk critical errors.   |
                        ;*-------------------------------------*
                        CMP     AH,23                           ; Was I (Ignore) Pressed?
                        JNE     Display_Critical_Error_140      ;       If no, then jump.
                        MOV     AX,IGNORE                       ; Load IGNORE into AX.
                        JMP     SHORT Display_Critical_Error_170  ; Jump to process return.
Display_Critical_Error_140:
                        CMP     AH,30                           ; Was A (Abort) pressed?
                        JNE     Display_Critical_Error_150      ;       If no then jump.
                        MOV     AX,ABORT ; Load ABORT into AX
                        JMP     SHORT Display_Critical_Error_170  ; Jump to process return.
Display_Critical_Error_150:
                        CMP     AH,19                           ; Was R (Retry) pressed?
                        JNE     Display_Critical_Error_160      ;       If no, then jump.
                        MOV     AX, RETRY                       ; Load RETRY into AX.
                        JMP     SHORT Display_Critical_Error_170  ; Jump to process return.
Display_Critical_Error_160:
                        CMP     AH,33     ; Was F (Fail) pressed?
                        JNE     Display_Critical_Error_120      ;       If no, jump to sound beep
                                                                ;       and get another key.
                        MOV     AX, FAIL  ; Load FAIL into AX.
Display_Critical_Error_170:
                        ;*-------------------------------------*
                        ;|      Restore the screen             |
                        ;*-------------------------------------*
                        PUSH    AX                              ; Save the critical error processing code.
                        MOV     AX,Video_Segment                ; Make the video segment our Data Segment.
                        MOV     ES,AX                           ;
                        MOV     DI, BX                          ; Point to the upper left box corner.
                        MOV     SI, OFFSET Save_Buffer          ; Point to our storage buffer.
                        MOV     CX,4                            ; Load number of lines to save.

Display_Critical_Error_180:
                        PUSH    CX                              ; Save line count.
                        MOV     CL,DL                           ; Load Box width into CL.
                        XOR     CH,CH                           ; Clear the high order byte.
                        PUSH    DI                              ; Save pointer to beginning of line.
                        REP     MOVSW                           ; Save the line in the buffer.
                        POP     DI                              ; Restore pointer to beginning of line.
                        ADD     DI,160                          ; Point to the next line.
                        POP     CX                              ; Restore the line count.
                        LOOP    Display_Critical_Error_180      ; Continue until done.
                        POP     AX                              ; Restore the critical error processing code.
                        CMP     _Prompt_Disk,0                  ; Are we prompting for a new disk?
                        JE      Display_Critical_Error_190      ;       If no, jump.
```

```
                        MOV     AX,RETRY                ; Signal retry.
Display_Critical_Error_190:
                        POP     ES                      ; Restore Registers.
                        POP     SI
                        POP     DI
                        POP     DX
                        POP     CX
                        POP     BX

                        POP     BP
                        RET
_Display_Critical_Error ENDP

                        END
```

C. MAKE file for RIA Program

```
#*-------------------------------------------------------*
# | Make file for the RIA Program Modules.              |
#*-------------------------------------------------------*
# | Works using Borland's Make program.                 |
# | To Compile RIA.EXE shell, you must have the         |
# |   SpontaneousAssembly Library Version 2.0           |
# |   or greater (Base Two Development, 11 East 200 North,  |
# |   OREM, Utah 84057 - 801-222-9500)                  |
# |                                                     |
#*-------------------------------------------------------*
.autodepend
# Path to C libraries.
LPATH=d:\c\lib
#
#  Defines for Assembly Modules.
#  RIA.EXE uses Spontaneous Assembly Library, version 2.0
#
ASM = D:\C\BIN\TASM
ASM_INCLUDE_PATH = D:\MASM\INCLUDE
ASM_LPATH = .\LIB
ASM_LIB = SAS.LIB
ASM_START = .\LIB\STARTs.OBJ
ASM_SWITCHES = /mx /D_model=small
/I$(ASM_INCLUDE_PATH)

RIA_OBJS = RIA.OBJ
RIADESC_OBJS = RIADESC.OBJ
RIALIB_OBJS = 123.obj allocat.obj beep.obj die.obj drwbox.obj
getunam.obj \
        harderr.obj inputstr.obj pnewdisk.obj readdesc.obj \
        savedesc.obj scrnsu.obj strdel.obj strins.obj \
        winbox.obj winrest.obj winsav.obj \
        ststore.obj getfile.obj centext.obj overwrit.obj \
        copyfile.obj critical.obj

#
#  C modules compile using Borland C/C++ Version 3.1
#
SWITCHES= -ms -w -N -c
# Add -v for source level debugging
CC = BCC

#
#  Compiles using PowerBasic Compiler, version 3
#
BASCOMP=C:\POWERBAS\PBC
# Directory for BASIC include files.
BASINC=

default: rialib.lib ria.exe fixit.exe riacopy.exe riadesc.exe \
        riadir.exe riakill.exe

#
# Implicit Rules
#
.c.obj:
$(CC) $(SWITCHES) $*.c

.asm.obj:
$(ASM)  $(ASM_SWITCHES) $*.asm

#
#  Explicit Rules
#
riacopy.exe: riacopy.obj rialib.lib
tlink @&&|
$(LPATH)\c0s+
$**
$*

$(LPATH)\cs+
rialib
|

riadir.exe: riadir.obj rialib.lib
tlink @&&|
$(LPATH)\c0s+
$**
$*

$(LPATH)\cs+
rialib
|

riakill.exe: riakill.obj rialib.lib
tlink @&&|
$(LPATH)\c0s+
$**
$*

$(LPATH)\cs+
rialib
|

fixit.exe: fixit.bas
$(BASCOMP) -DS$(BASINC) -ce fixit.bas

ria.exe: $(RIA_OBJS)
tlink @&&|
$(ASM_LPATH)\STARTS.OBJ+
$**
$*
```

```
$(ASM_LPATH)\$(ASM_LIB)
|

riadesc.exe: $(RIADESC_OBJS) rialib.lib
tlink @&&|
$(LPATH)\c0s+
$**
$*

$(LPATH)\cs+
rialib
|

lscdata.exe: lscdata.obj  comhndlr.obj rialib.lib
tlink @&&|
$(LPATH)\c0s+
$**
$*

rialib+
$(LPATH)\cs+
$(LPATH)\EMU+
$(LPATH)\MATHS
|


rialib.lib: $(RIALIB_OBJS)
&tlib rialib -+$?
#
#           End of Make File : RIA.MAK
#-------------------------------------------
```

D. Batch files to run RIA Program

1. *RIA.BAT - Main Batch File to Run RIA Program*

```
@ECHO OFF
REM *-----------------------------------------*
REM |    RIA.BAT - RIA Program version 1.1     |
REM *-----------------------------------------*
REM |    This batch file runs the RIA program.  This 'shell' is  |
REM |    required because our machine needs to reboot to         |
REM |    activate the external 5-1/4" 360K floppy drive.         |
REM |    The internal 5-1/4" drive make 360K disks               |
REM |    unreadable.                                             |
REM |    This file should not be in the RIA Programs             |
REM |    directory and on the PATH.                              |
REM |                                                            |
REM |    This Batch file requires the 4DOS Command Shell         |
REM |    which is a shareware command shell that can be          |
REM |    found at:                                               |
REM |    ftp://simtel.net/simtelnet/msdos/4dos                   |
REM *-----------------------------------------*
c:\lotus\lsc\programs\ria.exe
IF ERRORLEVEL=15 GOTO Start
REM *-----------------------------------------*
REM |    If RIA.EXE did not return 15, then we don't need        |
REM |    to reboot, so we are done.                              |
REM *-----------------------------------------*
ECHO RIA data processing completed.
QUIT
:Start
REM *-----------------------------------------*
REM |    Draw the main window (Full Screen)    |
REM *-----------------------------------------*
DRAWBOX 0 0 24 79 1  bright cyan on blue fill blue
REM *-----------------------------------------*
REM |    Put the title on the window.          |
REM *-----------------------------------------*
SCREEN 0 30
COLOR bright cyan on blue
ECHO %@CHAR[180]
SCREEN 0 31
COLOR bright yellow on blue
ECHO Steroid RIA Program
COLOR bright cyan on blue
SCREEN 0 50
ECHO %@CHAR[195]
REM *-----------------------------------------*
REM |    Draw the reboot window, and display message  |
REM *-----------------------------------------*
DRAWBOX 4 8 6  72 1 bright white on red fill red
SCREEN 5 10
COLOR bright white on red
ECHO  The machine needs to reboot to use the 360K external
      drive.
REM *-----------------------------------------*
REM |    Draw the instructions window, and fill it in  |
REM *-----------------------------------------*
DRAWBOX  9 2 13 77 2 red on white fill white
SCREEN 10 5
COLOR Blue on white
ECHO 1) Please turn on the external drive. (The switch is on the
      left rear).
SCREEN 12 5
ECHO 2) Push 'Y' to run RIA program.  (Push 'N' to abort.)
REM SCREEN 14 5
:Get_Key
REM *-----------------------------------------*
REM |  Get the user's input (y' or 'n)         |
REM *-----------------------------------------*
SCREEN 12 75
INKEY %%temp
IF '%temp%'=='n' GOTO Abort
IF '%temp%'=='y' GOTO Set_360
REM *-----------------------------------------*
REM |    Invalid Key pressed.  Beep then wait for  |
REM |    another key                               |
REM *-----------------------------------------*
BEEP
GOTO Get_Key
:Set_360
REM *-----------------------------------------*
REM |    Set up system to reboot to activate   |
REM |    the external 360K drive               |
REM *-----------------------------------------*
C:
CD \System
REM *-----------------------------------------*
REM |    Copy 360K CMOS system configuration into  |
REM |    file used by ROM.COM.                     |
REM *-----------------------------------------*
COPY CMOS_360.DAT CMOSROM.DAT       >NUL
REM *-----------------------------------------*
REM |    Copy the new AUTOEXEC.BAT to run the RIA  |
REM |    program after rebooting.  A copy of the normal  |
REM |    AUTOEXEC.BAT file must be in the C:\SYSTEM       |
REM |    directory and it will be overwritten and lost    |
REM |    otherwise.                                       |
REM *-----------------------------------------*
COPY AUTORIA.BAT \AUTOEXEC.BAT      >NUL
REM *-----------------------------------------*
REM |  Tell ROM.COM to load the 360K system configuration  |
```

```
REM |. in CMOSROM.DAT in CMOS, then reboot.                    |
REM |          switches: -r  Read data from CMOSROM.DAT        |
REM |                        and store in CMOS.                |
REM |                    -b  Reboot computer when done.        |
REM |  ROM.COM can be obtained from the SIMTEL archive:        |
REM |     ftp://simtel.net/simtelnet/msdos/sysutl/rom2.zip     |
REM *-----------------------------------------------------------*
ROM -r -b
REM *-----------------------------------*
REM |     Should never get to this point. |
REM *-----------------------------------*
COLOR bright cyan on blue
ECHO Program C:\SYSTEM\ROM.COM is apparently not
     present:
ECHO      Find another copy or obtain via ftp from
ECHO      ftp://simtel.net/simtelnet/msdos/sysutl/rom2.zip
COLOR bright white on blue
BEEP
QUIT
:abort
REM *-----------------------------------*
REM |     Abort program.  Do not reboot.  |
REM *-----------------------------------*
COLOR bright white on blue
CLS
COLOR blink bright red on blue
ECHO RIA program aborted by user.
COLOR bright white on blue
BEEP
QUIT
```

2. AutoRIA.BAT - Autoexec.Bat File Used When Rebooting to Initialize 360K Drive

```
@ECHO OFF
REM *-----------------------------------------------------------*
REM |  AUTORIA.BAT - RIA Program version 1.1                     |
REM *-----------------------------------------------------------*
REM |  This batch file is renamed to AUTOEXEC.BAT after by      |
REM |  RIA.BAT and is run after the machine reboots.            |
REM |                                                           |
REM |  This program should be located in C:|SYSTEM             |
REM |                                                           |
REM |  This Batch file requires the 4DOS Command Shell          |
REM |  which is a shareware command shell that can be           |
REM |  found at:                                                |
REM |  ftp://simtel.net/simtelnet/msdos/4dos                    |
REM *-----------------------------------------------------------*
REM *-----------------------------------*
REM |  Clear the screen and set the border color.  |
REM *-----------------------------------*
C:\DOS\Video Clear Border
C:\DOS\COLORReg 00 00 43 00
REM *-----------------------------------*
REM |  Set the keyboard repeat rate.      |
REM *-----------------------------------*
C:\DOS\FastKey /G /1
ECHO System Setup:
ECHO      Keyboard Speed Set.
REM *-----------------------------------*
REM |  Turn the number lock off.          |
REM *-----------------------------------*
C:\DOS\NUMOFF
ECHO      Number Lock turned off.
REM *-----------------------------------*
REM |  Load the system aliases.           |
REM *-----------------------------------*
ALIAS /R C:\System\Alias.Dat
ECHO      System aliases loaded.
REM *-----------------------------------*
REM |  Set up the ANSI keyboard remapping.  |
REM *-----------------------------------*
C:\DOS\ANSI ~{0;134;'C:\Batch\Menu.Bat';13p
C:\DOS\ANSI ~{0;133;'C:\Batch\tools FFIND 'p
ECHO      ANSI Keyboard assignments loaded.
C:\DOS\Video on
REM *-----------------------------------*
REM |  Switch to the C: disk and Users directory.  |
REM *-----------------------------------*
C:
CD C:\USERS
C:\DOS\Video clear border
REM *-----------------------------------*
REM |  Load type ahead buffer.            |
REM *-----------------------------------*
C:\SYSTEM\VISITYPE.COM
REM *-----------------------------------*
REM |  Load the screen saver with a RAM wedge.  |
REM *-----------------------------------*
C:\DOS\RamLoad ScrnSave.Com
C:\DOS\ScrnSave /D 3 -63
C:\DOS\video Clear Border
REM *-----------------------------------*
REM |  Set the environment variables, path and prompt.  |
REM *-----------------------------------*
PROMPT $E[0;30;46m$P$G$E[1;37;44m
SET PATH=C:\4dos;C:\DOS;C:\BATCH;C:\NORTON;C:\Win3;
     C:\Lotus;C:\NC;C:\WORD;
SET TEMP=C:\WIN3\TEMP
C:\4DOS\Palette North
REM *-----------------------------------*
REM |  Run the menu program.              |
REM *-----------------------------------*
C:
CD \Users
\system\runria.bat
```

3. RUNRIA.BAT - Runs RIA Program after Rebooting to use 360K Drive.

```
@ECHO OFF
REM *-----------------------------------------------------------*
REM |  RUNRIA.BAT - RIA Program version 1.1                      |
REM *-----------------------------------------------------------*
REM |  This batch file is launched from AUTOEXEC.BAT after      |
REM |  rebooting to use the 360K external drive.                |
REM |                                                           |
REM |  This program should be located in C:|SYSTEM             |
REM |                                                           |
REM |  This Batch file requires the 4DOS Command Shell          |
REM |  which is a shareware command shell that can be           |
REM |  found at:                                                |
REM |  ftp://simtel.net/simtelnet/msdos/4dos                    |
REM *-----------------------------------------------------------*
CD c:\lotus
REM *-----------------------------------*
REM |  Run the RIA program.               |
REM *-----------------------------------*
c:\lotus\lsc\programs\ria.exe
REM *-----------------------------------------------------------*
REM |  Prompt the user to turn off the External Drive           |
REM |  before we reboot.                                        |
```

```
REM *-------------------------------------------*
DRAWBOX 0 0 24 79 1 magenta on cyan fill cyan
DRAWBOX 3 5 5  75 1 yellow on blue fill blue
COLOR BRI WHITE ON BLUE
SCREEN 4 7
ECHO The machine needs to reboot to return to its normal
     configuration.
DRAWBOX  9 2 13 77 2 white on red fill red
SCREEN 10 5
COLOR BRI WHITE ON RED
ECHO 1) Please turn off the external drive. (The switch is on the
     left rear).
SCREEN 12 5
ECHO 2) Push 'Y' to continue.
:Get_Key2
INKEY %%temp
IF "%temp%"=="y" GOTO Set_12M
BEEP
GOTO Get_Key2

REM *-------------------------------------------*
REM |  Load the 1.2M ROM BIOS Data and reboot.   |
REM *-------------------------------------------*
:Set_12M
C:
CD \System
COPY CMOS_12M.DAT CMOSROM.DAT       >NUL
COPY autoexec.bat \autoexec.bat  >NUL
ROM -r -b
```

E. RIA.EXE - Shell Program run from RIA.BAT.

```
;*-------------------------------------------------*
;| FILE: RIA.ASM                                   |
;*-------------------------------------------------*
;| DESC: Shell to run the various RIA program modules.  |
;|                                                 |
;| NOTES: Version 1.1                              |
;|                                                 |
;| To Compile RIA.EXE shell, you must have the     |
;|   SpontaneousAssembly Library Version 2.0       |
;|   or greater (Base Two Development, 11 East 200 |
;|   North,Orem, Utah 84057 - 801-222-9500)        |
;*-------------------------------------------------*


include     MODEL.INC
include     CONSOLE.INC
include     WINDOW.INC
include     ECODES.INC
include     FILEIO.INC


.codeseg    lib
;*-------------------------*
;|   External Procedures.  |
;*-------------------------*
.extrn      start:auto, exit:auto, exit_ok:auto
.extrn      cget_chr:auto, console_init:auto, cput_str:auto
.extrn      near_init:auto, near_malloc:auto
.extrn      restore_screen:auto
.extrn      save_screen:auto, screen_bufsize:auto
.extrn      set_80cols:auto, set_backdropbuf:auto
.extrn      win_bufsize:auto, win_create:auto
.extrn      win_init:auto, win_move:auto
.extrn      goto_xy:auto
.extrn      get_chr:auto
.extrn      exec_prog:auto
.extrn      win_open:auto
.extrn      shrink_prog:auto
.extrn      sound_beep:auto
.extrn      win_close:auto
.extrn      win_select:auto
.extrn      clr_region:auto
.extrn      ch_dir:auto
.extrn      set_drive:auto
.extrn      set_directvideo:auto
.extrn      open_h:auto
.extrn      close_h:auto
.extrn      Remove_F:auto
.extrn      read_h:auto
.extrn      write_h:auto
.ends

.dataseg
;*-----------------*.
;|   Local Data    |
;*-----------------*
Option1         DB  '1)    Run a complete RIA (Dump data, do '
                DB  'calculations and',0
Option1a        DB  '      delete files)',0
len1            EQU $ - Option1 - 1
Option2         DB  '2)    Just transfer data from floppy.',0
len2            EQU $ - Option2 - 1
Option3         DB  '3)    Transfer data from floppy and do some '
                DB  'calculations',0
len3            EQU $ - Option3 - 1
Option3a        DB  '      (No files deleted)',0
```

```
len3a           EQU $ - Option3a- 1
Option4         DB  '4)    Do calculations only (No data from '
                DB  'data  transferred)',0
len4            EQU $ - Option4 - 1
Option4a        DB  '      (No files deleted)',0
len4a           EQU $ - Option4a - 1
Option5         DB  '5)    Do calculations and delete files.',0
Option5a        DB  '      (No data from data transporter)',0
len5            EQU $ - Option5 - 1
Option6         DB  '6)    Unload the data transporter. '
                DB  '[Obsolete option]',0
Num_Selections  EQU 6
Option_Table    DW  Option1, Option1a,Option2, Option3
                DW  Option3a, Option4, Option4a
                DW  Option5, Option5a, Option6
Num_Options     EQU ($ - Option_Table) /2
edit_prompt     DB  'Do you want to edit any of the data files'
                DB  '(y/n)?',0
len_edit        EQU $ - edit_prompt - 1
Prompt          DB  'Press the number (1-6) for you desired'
                DB  'options:',0
lenp            EQU $ - Prompt
Prompt_Handle   DW  ?
RIAVersion      DB  'RIA.exe  - Version 1.1'

scrbuf_addr     DW  ?
Menu_Handle     DW  ?
Edit_Handle     DW  ?
Memory_Error    DB  0DH,0AH,'Error while allocating '
                DB  'memory.',0DH,0AH,0
Selection       DB  ?
Lotus_Dir       DB  'C:\LOTUS',0
Time            DB  'C:\LOTUS\RIADATA\Timestam.p',0
Return_Code     DB  0
Lotus_Keys      DB  '/', 'f', 'r', 'a', 'u', 't', 'o', '1', '2', '3', '', 'r', 'i','a', 0
edit_window     window < (80-len_edit-4)/2,(25-3)/2,len_edit+4,3,BLUE*16+WHITE,,,,,,,,,,,,>
win             window <32,8,18,6,47h,2,2,1,2,,,,,,,,WCURSOR_OFF>
menu_window     window < (80-len4-4)/2,(25-Num_Options)/2,len4+4,Num_Options + 2,RED*16+WHITE,,,,,,,,,,,,>
Prompt_Window   window    < (80-lenp-4)/2,2,lenp+4,3,LTGRAY*16+RED,,,,,,,,,,,,>


Command_Line            DB  128 DUP(0)
Command_Unload          DB  'c:\lotus\lsc\programs\unload.com',0
Command_Convert         DB  'c:\lotus\lsc\programs\nounload.exe',0
Empty_Command_Line      DB  0
Command_RIAdesc         DB  'c:\lotus\lsc\programs\riadesc.exe',0
Command_RIAcopy         DB  'c:\lotus\lsc\programs\riacopy.exe',0
RIAcopy_Command_Line    DB  0
Command_123             DB  'c:\lotus\123.exe',0
Command_Line_123        DB  0
Command_RIAFix          DB  'c:\lotus\lsc\programs\fixit.exe',0
RIAfix_Command_Line     DB  0
Command_RIAsort         DB  'c:\lotus\lsc\programs\riadir.exe',0
RIAsort_Command_Line    DB  0
Command_RIAkill         DB  'c:\lotus\lsc\programs\riakill.exe',0
RIAkill_Command_Line    DB  0
RIA_Run_Code            DB  'C:\lotus\lsc\programs\Run_Code.RIA',0
.ends


IF NOT __TINY__
.stackseg
db 1024 dup(?) ;define a 1024 byte stack
.ends
ENDIF

;*-------------------------------------------------------------------*
;|    PROCEDURE:    MAIN                                             |
```

```
;|-------------------------------------------------------------------|
;|  DESC:     Main body of program.                                  |
;|                                                                   |
;|  IN:   DX segment address of PSP                                  |
;|                                                                   |
;|  ASSUMES: DS,ES   @DATASEG (same as @CODESEG    in TINY model)    |
;|           SS      @STACKSEG (same as @CODESEG in TINY model)     |
;*-------------------------------------------------------------------*

.codeseg

IF __TINY__
assume   cs:@codeseg, ds:@dataseg, es:@dataseg, ss:@dataseg
ELSE
assume   cs:@codeseg, ds:@dataseg, es:@dataseg, ss:@stackseg
ENDIF

.public  main
.proc    main        auto

         MOV         AX,@dataseg
         MOV         ES,AX
         MOV         DS,AX

         CALL        Initialize
         CALL        shrink_prog
         CALL        Time_Stamp
         MOV         SI,OFFSET RIA_Run_Code  ; Point to the file which contains reboot info.
         MOV         AX,O_RDONLY             ; Set attribute to read only.
         .call       open_h                  ; Open the file.
         JC          main_10                 ; If can't open the file, then doesn't exist.
         MOV         CX,1                    ; Ready 1 byte to read.
         MOV         SI,OFFSET Selection     ; Point to the selection code.
         MOV         Return_Code,0FFH        ; Load Return code to signal reboot.
         .call       read_h                  ; Read the selection.
         JE          main_5                  ; If just rebooted, then jump.
         MOV         Return_Code,0           ; Set Return code for no need to reboot.
main_5:
         .call       close_h                 ; Close the code file.
         MOV         SI,OFFSET RIA_Run_Code  ; Point to the code file name.
         .call       remove_f                ; Delete the code file.
         CMP         Return_Code,0FFH        ; If we just rebooted, then jump over rest.
         JE          main_25
main_10:
         CALL        Draw_Menu
         CALL        Get_Selection
         MOV         BX,Menu_Handle
         CALL        win_close
         CMP         Selection,6
         JNE         main_20
         CALL        Dump_DTP
         MOV         DI,scrbuf_addr          ;ES:DI->screen buffer address
         CALL        restore_Screen
         JMP         SHORT main_10
main_20:
         CALL        Test_Reboot
main_25:
         ;
         ; If requested, copy files from floppy.
         ;
         CALL        RIACopy
         MOV         DI,scrbuf_addr          ;ES:DI->screen buffer address
         CALL        restore_screen
         ;
         ; Sort the Data files.
         ;
         CALL        RIAsort
         MOV         DI,scrbuf_addr          ;ES:DI->screen buffer address
```

```
            CALL    restore_screen
            ;
            ; If required, run the Fixit program.
            ;
            CALL    RIAfix
            MOV     DI,scrbuf_addr          ;ES:DI->screen buffer address
            CALL    restore_screen
            ;
            ; Sort the Data files.
            ;
            CALL    RIAsort
            MOV     DI,scrbuf_addr          ;ES:DI->screen buffer address
            CALL    restore_screen
            ;
            ; Run Lotus
            ;
            CALL    Run_Lotus
            MOV     DI,scrbuf_addr          ;ES:DI->screen buffer address
            CALL    restore_screen
            ;
            ; Describe created worksheets
            ;
            CALL    RIAdesc
            MOV     DI,scrbuf_addr          ;ES:DI->screen buffer address
            CALL    restore_screen
            ;
            ; Kill data files if requested.
            ;
            CALL    RIAkill
            MOV     DI,scrbuf_addr          ;ES:DI->screen buffer address
            CALL    restore_screen          ;restore original screen
            ;
            ; Kill time stamp file.
            ;
            CALL    kill_stamp
            MOV     AL, Return_Code
            MOV     AH,4CH
            INT     21H
.endp main


.proc       Test_Reboot auto
            CMP     Selection,3
            JG      Test_Reboot_10
            MOV     AX, O_WRONLY OR O_CREAT OR O_TRUNC; Set file open attributes.
            MOV     SI, OFFSET RIA_Run_Code         ; Point to the RIA run code file name.
            .call   open_h                          ; Create/Truncate it.
            MOV     CX,1                            ; Load one byte to write.
            MOV     SI,OFFSET Selection             ; Point to our selection.
            .call   write_h                         ; Write the byte.
            .call   close_h                         ; Close the code file.
            MOV     AH,4CH                          ; Load DOS function to exit with Return code.
            MOV     AL,0FH                          ; Load Return code.
            INT     21H                             ; Exit
Test_Reboot_10:
            RET                                     ; Don't need to reboot.
.endp       Test_Reboot

.proc       RIAcopy  auto
            CMP     Selection,3
            JG      RIAcopy_10

            MOV     DI,OFFSET Command_RIAcopy
            MOV     SI,OFFSET RIAcopy_Command_Line
            CALL    exec_prog
RIAcopy_10:
            RET
.endp       RIAcopy
```

```
;* ---------------------------------------------------------------- *
;| Procedure: RIAdesc                                               |
;|------------------------------------------------------------------|
;| This program runs the RIA worksheet description program.         |
;* ---------------------------------------------------------------- *
.proc   RIAdesc auto

        CMP     Selection,2                 ; Are we just unloading the data?
        JE      RIAdesc_1                   ; If yes, then nothing to describe?
        CMP     Selection,6                 ; Are we doing an old data dump?
        JE      RIAdesc_1                   ; If yes, then nothing to describe.
        MOV     DI,OFFSET Command_RIAdesc   ; Point to the RIA Description command line.
        MOV     SI,OFFSET Empty_Command_Line ; Give it an empty command line.
        CALL    exec_prog                   ; Run the program.
RIAdesc_1:
        RET
.endp   RIAdesc


.proc   Time_Stamp      auto
        MOV     SI,OFFSET Time
        MOV     AX, O_RDWR OR O_TRUNC OR O_CREAT
        CALL    Open_H
        CALL    Close_H
        RET
.endp   Time_Stamp

.proc   Kill_Stamp auto
        MOV     SI,OFFSET Time
        CALL    Remove_F
        RET
.endp   Kill_Stamp


.proc   Dump_DTP auto
        MOV     DI,OFFSET Command_Unload
        MOV     SI,OFFSET Empty_Command_Line
        CALL    exec_prog
        MOV     DI,OFFSET Command_Convert
        MOV     SI,OFFSET Empty_Command_Line
        CALL    exec_prog
Dump_DTP_10:
        RET
.endp   Dump_DTP


.proc   RIAsort auto
        MOV     DI,OFFSET Command_RIAsort
        MOV     SI,OFFSET RIAsort_Command_Line
        CALL    exec_prog
RIAsort_10:
        RET
.endp   RIAsort


.proc   RIAkill auto
        CMP     Selection,1
        JE      RIAkill_5
        CMP     Selection,5
        JNE     RIAkill_10
RIAkill_5:
        MOV     DI,OFFSET Command_RIAkill
        MOV     SI,OFFSET RIAkill_Command_Line
        CALL    exec_prog
RIAkill_10:
        RET
```

```
.endp       RIAkill


.proc       Run_Lotus auto
            CMP       Selection,2
            JE        Run_Lotus_10
            CMP       Selection,6
            JE        Run_Lotus_10
            MOV       SI,OFFSET @dataseg:Lotus_Dir
            MOV       AL,BYTE PTR [SI]
            CALL      set_drive
            CALL      ch_dir
            MOV       DI,OFFSET Command_123
            MOV       SI,OFFSET Command_Line_123
            CALL      exec_prog
Run_Lotus_10:
            RET
.endp       Run_Lotus


.proc       RIAfix auto
            CMP       Selection,2
            JE        RIAfix_10
            CMP       Selection,6
            JE        RIAfix_10
            MOV       BX,@dataseg:Edit_Handle
            CALL      win_open
            CALL      win_select
            XOR       AX,AX
            INC       AL
            CALL      goto_xy
            MOV       SI,OFFSET @dataseg:Edit_Prompt
            CALL      cput_str
RIAfix_5:
            CALL      get_chr
            CMP       AL,'y'
            JE        RIAfix_8
            CMP       AL,'Y'
            JE        RIAfix_8
            CMP       AL,'n'
            JE        RIAfix_10
            CMP       AL,'N'
            JE        RIAfix_10
            MOV       AX,110
            MOV       DX,10
            CALL      sound_beep
            JMP       RIAfix_5
RIAfix_8:
            MOV       BX,@dataseg:Edit_Handle
            CALL      win_close
            MOV       DI,OFFSET Command_RIAfix
            MOV       SI,OFFSET RIAfix_Command_Line
            CALL      exec_prog
RIAfix_10:
            MOV       BX,@dataseg:Edit_Handle
            CALL      win_close
            RET
.endp       RIAfix


.proc       Get_Selection   auto
Get_Selection_10:
            CALL      get_chr
            JC        Get_Selection_20
            CMP       AL,'1'
            JL        Get_Selection_20
            CMP       AL,'1'+ Num_Selections
```

```
            JGE     Get_Selection_20
            SUB     AL,'0'
            MOV     Selection,AL
            RET
Get_Selection_20:
            MOV     AX,110
            MOV     DX,10
            CALL    sound_beep
            JMP     SHORT Get_Selection_10
.endp       Get_Selection


.proc       Draw_Menu   auto
            MOV     BX,@dataseg:Prompt_Handle
            CALL    win_open
            CALL    win_select
            XOR     AX,AX
            INC     AL
            MOV     SI,OFFSET @dataseg:Prompt
            CALL    Goto_XY
            CALL    CPUT_STR
            MOV     BX,@dataseg:Menu_Handle
            CALL    win_open
            CALL    win_select
            MOV     AH,0
            MOV     AL,1
            MOV     CX,Num_Options
            XOR     BX,BX
Draw_Menu_10:
            MOV     SI,Option_Table[BX]
            PUSH    AX
            PUSH    BX
            CALL    Goto_XY
            CALL    CPUT_STR
            POP     BX
            ADD     BX,2
            POP     AX
            INC     AH
            LOOP    Draw_Menu_10
            RET
.endp       Draw_Menu

;* ---------------------------------------*
;|  Procedure:  Initialize                |
;* ---------------------------------------*
;|  Inititializes Console, windowing system,  |
;* ---------------------------------------*
.proc       Initialize auto
            MOV     AX,8000             ; ask for all available heap space
            CALL    near_init           ; init the near heap
            CALL    console_init        ; init the console i/o system
            CALL    set_directvideo
            CALL    screen_bufsize      ; get size of screen buffer
            CALL    near_malloc         ; error allocating screen buffer?
            JC      Initialize_70       ;       y: exit without screen restore
            MOV     scrbuf_addr,di      ;       n: save screen buffer address
            CALL    save_screen         ;       and save the screen
            CALL    set_80cols          ; force 80 column mode if not already
            MOV     AX,WWORKBUF_SIZE    ; window system work buffer size
            CALL    near_malloc         ; error allocating work buffer?
            JC      Initialize_70       ;       y: exit
            CALL    win_init            ;       n: init windowing system
            MOV     DI,scrbuf_addr      ; ES:DI->screen buffer address
            CALL    set_backdropbuf     ; use screen as backdrop
            MOV     SI,offset @dataseg:Menu_Window
            CALL    win_bufsize         ; get buffer size for WIN
            CALL    near_malloc         ; error allocating WIN buffer?
            JC      Initialize_70       ;       y: exit
```

```
            CALL    win_create                  ;       n: create WIN
            MOV     @dataseg:Menu_Handle, BX
            MOV     SI,offset @dataseg:Prompt_Window
            CALL    win_bufsize                 ; get buffer size for WIN
            CALL    near_malloc                 ; error allocating WIN buffer?
            JC      Initialize_70               ;       y: exit
            CALL    win_create                  ;       n: create WIN
            MOV     @dataseg:Prompt_Handle, BX
            MOV     SI,offset @dataseg:edit_window
            CALL    win_bufsize                 ; get buffer size for WIN
            CALL    near_malloc                 ; error allocating WIN buffer?
            JC      Initialize_70               ;       y: exit
            CALL    win_create                  ;       n: create WIN
            MOV     @dataseg:Edit_Handle, BX
            CALL    win_close                   ; Make sure window is closed.
Initialize_60:
            RET
Initialize_70:
            MOV     SI,OFFSET @dataseg:Memory_Error
            CALL    CPUT_STR
            MOV     AX,1
            CALL    EXIT
.endp       Initialize

.ends

; *-------------------------------------------------------------------------|
; |   Stack normalization and memory management initialization labels       |
; |                                                                         |
; |   NOTE: These declarations must remain after the declaration of the stack |
; |   and anything in the stack segment. These labels define the end of the |
; |   stack and the program, which is where the near and far heaps are placed |
; |   by default. These declarations do not affect the size of the program and |
; |   may be left here even if the stack is not normalized and the heaps are |
; |   not used.                                                             |
; *-------------------------------------------------------------------------|

.public     nheap_default, fheap_default
IF NOT __TINY__
.stackseg
IF __SMALL__ OR __MEDIUM__
.public     stack_end           ;used by START to normalize stack
.label      stack_end   word    ;must be defined past entire stack
ENDIF
.label      nheap_default   word ;used by the near heap
.label      fheap_default   word ;used by the far heap
.ends
ELSE
_BSSEND         segment     byte public 'STACK'
.label          nheap_default   word ;used by the near heap
.label          fheap_default   word ;used by the far heap
_BSSEND         ends
% @codeseggroup_BSSEND
ENDIF


            end   start         ;specify START as starting address
```

F. LSCDATA.EXE - Data Collection Program.

1. *Header File for LSCDATA.C*

```
#ifndef _LSCDATA_H_
#define _LSCDATA_H_
// *-------------------------------------------------------*
// | SOURCE FILE: LSCDATA.H                                |
// *-------------------------------------------------------*
// | Header file for LSCDATA.C - Data Collection           |
// |                                                       |
// | RIA program version 1.1                               |
// *-------------------------------------------------------*
// *-------------------------------------------------------*
// |          Prototypes for comhndlr.asm functions.       |
// *-------------------------------------------------------*
extern void Disable_INT_Handler( void );
```

```
extern void Enable_INT_Handler( void );
extern int  Display_Critical_Error( char * );
// *-------------------------------------------------*
// |                Function prototypes              |
// *-------------------------------------------------*
void Initialize_Video( void );
void Capture_Interrupts( void );
void Allocate_Buffers( void );
void Clear_Screen ( void );
void Move_To_Output_Buffer( void );
unsigned int Length_Data( unsigned int , unsigned int );
unsigned int Scan_for_NULL ( int );
void Display_Waiting ( void );
int Done( void );
void Signal_OK( void );
void Error_Exit (char *, char * );
void Write_Data( void );
int Get_Code( unsigned int );
void Move_Data_Up ( unsigned int );
void Reset_Timer( void );
time_t Elapsed_Time( void );
void Open_File(int , unsigned int ) ;
void No_NULL_Found ( unsigned int );
void Display_Bytes_Received( void );
int draw_box( int, int, int, int ) ;
void Display_Free( void );
void promptForNewDisk(void);
// *-------------------------------------------------*
// |                Global Constants                 |
// *-------------------------------------------------*
#define TRUE -1
#define FALSE 0
#define INPUT_UL_X 4
#define INPUT_UL_Y 4
#define INPUT_LR_X 22
#define INPUT_LR_Y 22
#define IGNORE 0
#define RETRY 1
#define ABORT 2
// *-------------------------------------------------*
// |              Data from Prompt for Disk          |
// *-------------------------------------------------*
#define RESET_DRIVE 0
#define VERIFY_SECTORS 4
#define HEAD 0
#define DISK 0
#define TRACK 0
#define SECTOR 1
#define NUMBER_SECTORS 1
#define BOX_UL_X 6
#define BOX_UL_Y 11
#define BOX_LR_X 73
#define BOX_LR_Y 13
// *-----------------------------*
// |  Data for Waiting for data  |
// *-----------------------------*
#define Wait_X1 9
#define Wait_X2 9+64
#define Wait_Y1 9
#define Wait_Y2 9+4
#endif
```

2. LSC Raw Data Collection Program - LSCDATA.

```
// *-----------------------------*
// | SOURCE FILE: LSCDATA.C      |
// *-----------------------------*
// | Collects data from the Beckman LSC  |
// *-------------------------------------*
#include <stdlib.h>
#include <stdio.h>
#include <dos.h>
#include <dir.h>
#include <conio.h>
#include <time.h>
#include <string.h>
#include <bios.h>
#include <ctype.h>
#include "lscdata.h"
#include "rialib.h"
// *-------------------------------------------------*
// |          Program - LSCdata.Exe (Version 1.2)    |
// *-------------------------------------------------*
// | Gets data from the Beckman LSC, extracts the needed info, |
// | and stores                                      |
// | it to disk. To try to minimize data loss due to power |
// | outages, the data is stored after 15 second waits, and the |
// | file is closed until the next data is ready.    |
// |                                                 |
// | Needs lscdata.h and comhndlr.asm.               |
// |                                                 |
// | Compiled using Borland C/C++ 3.1.               |
// |                                                 |
// | Version 1.1 - Fixed bug during wrapping of COM buffer. |
// | Version 1.2 - Try to fix mysterious bug where file start |
// |               code (200,BE) gets repeated causing a user |
// |               number of 20.                     |
// *-------------------------------------------------*
#ifdef DEBUG
#       include <fcntl.h>
#       include <sys\stat.h>
        int Input_Handle;
        FILE *debug_file;
#endif

#ifdef DEBUG
#       define WAIT 0
#else
#       define WAIT 15
#endif

#define BUFFER_SIZE 10240

char  *COM_Buffer,
      *Output_Buffer,
      Output_Path[80]="",
      COM_Data_Pending=0,
      Output_Data_Waiting=0,
      Output_File_Name[13]="",
      Prompt_Disk=0,
      Assay_Name[13];

int   Page,
      Columns,
      Mode,
      User_Number=0,
      Use_Code[14]={ 0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
      Window_Line=1;
unsigned char   Normal_Attrib;
unsigned int    COM_Head=0,
                COM_Tail=0,
                COM_Errors=0,
                REC_Code=0,
                REC_Sub_Code=0,
                Cluster_Size,
```

```
                Records,
                UNK_Code=0,
                REC_Records=0,
                Check_Sum_Error,
                Output_Tail=0;
unsigned long   Bytes_Received=0,
                Bytes_Free;
struct diskfree_t Disk_Data;
#ifdef DEBUG
int Drive=1;
#else
int Drive=0;
#endif
char Disk_Drive[3]="a:";
char far *Screen;
time_t Last_Time=0;
FILE *Output_File;

// *-------------------------------------------------*
// | PROCEDURE: main                                 |
// *-------------------------------------------------*
// | PURPOSE: Main module of RIA data collection program |
// | ON CALL: Nothing.                               |
// | RETURNS: Nothing                                |
// *-------------------------------------------------*
void
main() {
    unsigned defaultdrive;

    _dos_getdrive( &defaultdrive );
    if ( Drive == 0 ) Drive = defaultdrive;
    if ( Drive != 1 )
        Disk_Drive[0] = 'a' + Drive - 1;
    Output_File = NULL;
    Allocate_Buffers();
    Capture_Interrupts();
    /**-------------------------------------**/
    /*| Shouldn't have to exit, except on error |*/
    /**-------------------------------------**/
    while ( TRUE ) {
        promptForNewDisk();
        Reset_Timer();
        /**-------------------------------**/
        /*| Continue until the F10 key is pressed |*/
        /**-------------------------------**/
        while ( !Done() ) {
            if ( COM_Data_Pending )
                {
                    Move_To_Output_Buffer();
                }
            else
                if ( Output_Data_Waiting )
                    {
                        if ( Elapsed_Time() > WAIT )
    {
                            Write_Data();
                            Reset_Timer();
                        }
                    }
        }
        Write_Data();
        // Waits until the F10 key is pressed again
        Display_Waiting();
    }
}
/* End Main */

// *-------------------------------------------------*
// | PROCEDURE: Display_Status                       |
// |-------------------------------------------------|
// | PURPOSE: Displays current status of current function on |
// |          screen                                 |
// | ON CALL: code - code for current action         |
// | RETURNS: Nothing                                |
// *-------------------------------------------------*
void
Display_Status( int Code ) {
    char Scratch[80];
    int x,y, Center;

    x = wherex();
    y = wherey();
    memset(Scratch,' ',45);
    memset(Scratch+45,0,1);
    gotoxy( 33,16 );
    cputs( Scratch );
    switch (Code) {
        case 0:
            strcpy( Scratch, 'Initializing' );
            break;
        case 200:
            strcpy( Scratch, 'Beginning new ' );
            if ( User_Number == 6 )
                strcat( Scratch, 'RECOVERY file.');
              else
                strcat( Scratch, Assay_Name);
            strcat( Scratch, ' file.');
            break;
        case 100:
            strcpy( Scratch, 'Extracting RECOVERY
                record.' );
            break;
        case 52:
            strcpy( Scratch, 'Extracting ');
            strcat( Scratch, Assay_Name );
            strcat( Scratch,' replicate average.');
            break;
        case 999:
            strcpy( Scratch, 'Data file closed. Waiting for
                data.' );
            break;
        default:
            strcpy( Scratch, 'Skipping unneeded data.' );
            break;
    }
    Center = 33 + (45 - strlen(Scratch) )/2;
    gotoxy( Center, 16 );
    cputs( Scratch );
    gotoxy( x, y );
}


// *-------------------------------------------------*
// | PROCEDURE: Display_Free                         |
// |-------------------------------------------------|
// | PURPOSE: Displays free space on the data disk.  |
// | ON CALL: Nothing                                |
// | RETURNS: Nothing                                |
// *-------------------------------------------------*
void
Display_Free() {
int x,y,i,j,l,p=19,s;
char Scratch[21];
x = wherex();
y = wherey();
gotoxy( 63, 23 );
```

```
sprintf( Scratch, "%10lu", Bytes_Free );
l = strlen( Scratch );
s = l-1;
Scratch[20] = 0;
for ( i=l-1; i>=0; i-3 ) {
        for (j = 0; j<3; j++ ) {
        if (Scratch[s] == ' ')
                {
                i =-1;
                j = 4;
                }
                else
                Scratch[p--] = Scratch[s--];
        }
        if ( i>=0 && Scratch[s] != ' ' )
        Scratch[p--] = ',';
}
for (i = 0; i <= p; i++)
        Scratch[i] = ' ';
cputs( Scratch + 10 );
gotoxy( x, y );
}
```

```
// *-------------------------------------------------*
// | PROCEDURE: Display_Filename                     |
// |-------------------------------------------------|
// | PURPOSE: Display the filename for file created. |
// | ON CALL: filename - name of file created.       |
// | RETURNS: Nothing                                |
// *-------------------------------------------------*
void
Display_Filename( char *filename ) {
int x, y;
x = wherex();
y = wherey();
window( INPUT_UL_X, INPUT_UL_Y, INPUT_LR_X,
INPUT_LR_Y );
gotoxy( 2, Window_Line );
cprintf("%s\n", filename );
gotoxy( x, y );
window( 1, 1, 80, 25 );
if (Window_Line != INPUT_LR_Y - INPUT_UL_Y + 1 )
        ++Window_Line;
Display_Free();
}
```

```
// *-------------------------------------------------*
// | PROCEDURE: Display_Num_Records                  |
// |-------------------------------------------------|
// | PURPOSE: Display the number of records in created file. |
// | ON CALL: Num_Records - Number of records in file. |
// | RETURNS: Nothing                                |
// *-------------------------------------------------*
void
Display_Num_Records( int Num_Records ) {
int x, y;
x = wherex();
y = wherey();
window( INPUT_UL_X, INPUT_UL_Y, INPUT_LR_X,
INPUT_LR_Y );
gotoxy( 16, Window_Line - 1 );
cprintf("%3d", Num_Records );
gotoxy( x, y );
window( 1, 1, 80, 25 );
Display_Free();
}
```

```
/*
 *-------------------------------------------------*
 | Procedure: Move_To_Output_Buffer                |
 |-------------------------------------------------|
 | Moves data from the COM buffer to the Output buffer as a |
 | series of NULL terminated strings.              |
 |-------------------------------------------------|
 | Parameters: (none)                              |
 | Returned: (none)                                |
 | Variables: unsigned int Head    COM buffer head on call. |
 |            unsigned int Tail    COM buffer tail on call. |
 |            unsigned int i       Loop counter.   |
 |            unsigned int Pointer Pointer into COM buffer. |
 |            unsigned char ch     Current character. |
 *-------------------------------------------------*
 */
void
Move_To_Output_Buffer() {
unsigned int Head,Tail,i, Pointer;
unsigned char ch;
// *-------------------------------------------------*
// | Disable interrupts while we copy current COM buffer |
// | head and tail, and clear the data pending flag. |
// *-------------------------------------------------*
disable();
Head = COM_Head;
Tail = COM_Tail;
COM_Head = COM_Tail;
COM_Data_Pending = 0;
enable();
for ( i=0; i < Length_Data( Head, Tail ); i++) {
        if ( (Head + i) == BUFFER_SIZE )
                Pointer = Head + i - BUFFER_SIZE;
        else
                Pointer = Head + i;
        ch = COM_Buffer[Pointer];
        Bytes_Received++;
        if ( ( ch < 32 && ch != 0x0A ) || ch>126 )
                /* If not normal character or linefeed */
                /* set ch to 0xFF so we won't move it */
                /* to the output buffer.              */
                ch = 0xFF;
        else
                switch ( ch ) {
                        case 0x0A:
                                // Set End of line to NULL so we
                                // can easily access them later.
                                ch = 0x00;
                                break;
                        default:
                                break;
                } /* End switch */
        // If it's a normal character, store it, else skip.
        if ( ch != 0xFF )
        Output_Buffer[Output_Tail++] = ch;
} /* End for loop */
Display_Bytes_Received();
++Output_Data_Waiting;
}
```

```
// *-------------------------------------------------*
// | PROCEDURE: Display_Bytes_Received               |
// |-------------------------------------------------|
// | PURPOSE: Display bytes recieved in current file so far. |
// | ON CALL: NOTHING                                |
// | RETURNS: NOTHING                                |
```

```
// *-------------------------------------------------*
void
Display_Bytes_Received() {
int x,y;
x = wherex();
y = wherey();
gotoxy( 68,2 );
printf( "%10lu", Bytes_Received );
gotoxy( x, y );
}

// *-------------------------------------------------*
// | PROCEDURE: Length_Data                          |
// |-------------------------------------------------|
// | PURPOSE: Returns the amount of data in the COM buffer. |
// | ON CALL: Head - Position of start of ring buffer |
// |          Tail - Position of end of ring buffer.  |
// | RETURNS: Number of bytes waiting in the COM buffer. |
// *-------------------------------------------------*
unsigned int
Length_Data( unsigned int Head, unsigned int Tail ) {
if ( Head < Tail )
      return ( Tail - Head );
      else
      return ( BUFFER_SIZE + Tail - Head  );
}

// *-------------------------------------------------*
// | PROCEDURE: Scan_for_NULL                        |
// |-------------------------------------------------|
// | PURPOSE: Find start of next line in COM buffer. |
// | ON CALL: Offset - Current position in buffer.   |
// | RETURNS:  Position of next line in COM buffer   |
// |           zero if line end not found.           |
// *-------------------------------------------------*
unsigned int
Scan_for_NULL ( int Offset ) {
      int i;

      for (i=Offset; i<Output_Tail; i++ ) {
            if (Output_Buffer[i] == 0 )
                  return ( i + 1 );
      }
      return ( 0 );
}


// *-------------------------------------------------*
// | PROCEDURE: Display_Waiting                      |
// |-------------------------------------------------|
// | PURPOSE: Display message when disk is to be removed to |
// |          dump data.                             |
// | ON CALL: Nothing                                |
// | RETURNS: Nothing                                |
// *-------------------------------------------------*
void
Display_Waiting () {
      char box_save [(Wait_X2 - Wait_X1 +1 ) * (Wait_Y2 -
            Wait_Y1+1 ) * 2 ];

      gettext( Wait_X1, Wait_Y1, Wait_X2, Wait_Y2, box_save );
      highvideo();
      draw_box( Wait_X1, Wait_Y1, Wait_X2, Wait_Y2 );
      gotoxy( Wait_X1+3, Wait_Y1+1 );
      _setcursortype(_NOCURSOR);
      gotoxy( Wait_X1+3, Wait_Y1+1 );
      cputs("Remove the disk from the drive and use RIA to copy
            the' );
      gotoxy( Wait_X1+3, Wait_Y1+2 );
      cputs("data onto the Zenith for processing.' );
      gotoxy( Wait_X1+3, Wait_Y1+4 );
      cputs("Press F10 when ready to start again.' );
      while ( !Done() ) {
      }
      puttext( Wait_X1, Wait_Y1, Wait_X2, Wait_Y2, box_save );
      _setcursortype(_NORMALCURSOR);
      lowvideo();
}

// *-------------------------------------------------*
// | Procedure:  Done                                |
// |-------------------------------------------------|
// | Checks to see if a key was pressed, and returns 0 if no key |
// | pressed.  If a key was pressed, and it wasn't F10, it beeps and |
// | returns 0 as well. If F10 was pressed, returns -1.  |
// |-------------------------------------------------|
// | Parameters: (none)                              |
// | Returned:     0     if F10 not pressed          |
// |              -1     if F10 pressed.             |
// | Variables:  (none)                              |
// *-------------------------------------------------*
int Done() {
#     define DoneX1 24
#     define DoneX2 56
#     define DoneY1 11
#     define DoneY2 13
      int keypressed,Bad_Key=1;
      char save_buf[(DoneX2-DoneX1+1) * (DoneY2-DoneY1+1)
* 2];
      if ( kbhit() ) {
      if ( (keypressed=getch()) != 0 )
            {
                  /* Non-extended key pressed, so make */
                  if (keypressed == 27 )
                        {
                        gettext( DoneX1, DoneY1, DoneX2,
                              DoneY2, save_buf );
                        highvideo();
                        draw_box( DoneX1, DoneY1, DoneX2,
                              DoneY2 );
                        gotoxy( 26, 12 );
                        cputs("Do you really want to abort?");
                        while ( Bad_Key ) {
                              keypressed = getch();
                              tolower(keypressed);
                              switch ( keypressed ) {
                              case 'n':
                                    Bad_Key = 0;
                                    break;
                              case 'y':
                                    Disable_INT_Handler();
                                    exit(0);
                                    break;
                              default:
                                    // a noise, then return 0.
                                    sound(110);
                                    delay(200);
                                    nosound();
                                    break;
                              }
                        }
                        puttext( DoneX1, DoneY1, DoneX2,
                              DoneY2, save_buf );
                        lowvideo();
                        }
                  else
```

```
                    // a noise, then return 0.
                    sound(110);
                    delay(200);
                    nosound();
                    return ( 0 );
                }
                else
                {
                    if ( getch() != 68 )
                      {
                        // An extended key was pressed, but it
                        // wasn't F10, so make a rude noise and
                        // return 0.
                        sound(110);
                        delay(200);
                        nosound();
                        return ( 0 );
                      }
                    else
                      {
                        if ( COM_Data_Pending ||
Output_Data_Waiting )
                          {
                            // F10 was pressed, but we but there
                            // is still data to process.  Make a
                            // rude noise and return 0.
                            sound(110);
                            delay(200);
                            nosound();
                            return ( 0 );
                          }
                        else
                            // F10 key pressed, so signal done.
                            return (-1);
                      }
}
}
return(0);                      /* If no key pressed, return 0 */
}


// *-------------------------------------------------------*
// | PROCEDURE: Get_Tokens                                 |
// |-------------------------------------------------------|
// | PURPOSE: Get the desired number of tokens from a string. |
// | ON CALL: String - String to extract tokens from       |
// |          Num_Tokens - Number of tokens to extract     |
// |          Token - Array to store tokens in.            |
// | RETURNS: -1 if everything OK or                       |
// |          zero if not enough tokens found.             |
// *-------------------------------------------------------*
int
Get_Tokens( char *String, int Num_Tokens, char *Token[] ) {
int i;
if (( Token[0] = strtok( String, "," )) == NULL )
      return (0);
for (i=1; i<Num_Tokens; i++ ) {
      if (( Token[i] = strtok( NULL, "," )) == NULL )
            return (0);
}
return (-1);
}

// *-------------------------------------------------------*
// | PROCEDURE: Error_Exit                                 |
// |-------------------------------------------------------|
// | PURPOSE: Prints an error message on the screen and exits. |
// | ON CALL:    Format - Formatting string, or Formatted error |
// |                                                  message. |
// |             Name - Single parameter to use with format.  |
// | RETURNS: Does not return.                             |
// *-------------------------------------------------------*
void
Error_Exit (char *Format, char *Name) {
window( 1, 1, 80, 25);
gotoxy( 1, 25 );
if ( Name == NULL )
            printf( Format );
      else
            printf( Format, Name );
      Disable_INT_Handler();
      exit (1);
}

// *-------------------------------------------------------*
// | PROCEDURE: Open_File                                  |
// |-------------------------------------------------------|
// | PURPOSE: Open recovery, unknown, or date file.        |
// | ON CALL: Code - file type to open.                    |
// |          Offset - Current offset in output buffer.    |
// | RETURNS: Nothing                                      |
// *-------------------------------------------------------*
void
Open_File(int Code, unsigned int Offset ) {
      int Day_Code, Month_Code,i;
      char *Data,U_Number[3]="";
      char *Wd,*Da,*Mo,*Yr, *Pointer;
      char *Abriev_Month[] = { 'JAN',
                               'FEB',
                               'MAR',
                               'APR',
                               'MAY',
                               'JUN',
                               'JUL',
                               'AUG',
                               'SEP',
                               'OCT',
                               'NOV',
                               'DEC' };
      char *Month[] = { 'January',
                        'February',
                        'March',
                        'April',
                        'May',
                        'June',
                        'July',
                        'August',
                        'September',
                        'October',
                        'November',
                        'December'};
      char *Abriev_Weekday[] = { 'MON',
                                 'TUE',
                                 'WED',
                                 'THU',
                                 'FRI',
                                 'SAT',
                                 'SUN'};
      char *Weekday[] = { 'Monday',
                          'Tuesday',
                          'Wednesday',
                          'Thursday',
                          'Friday',
                          'Saturday',
                          'Sunday' };
```

```
if ( Code == 1 ) {
        /**-------------------------------------**/
        /*|   This section opens a new recovery sub file.  |*/
        /**-------------------------------------**/
        // *-------------------------------------*
        // |   If this is not the recovery User Number (6)  |
        // |   then something has gone really wrong.        |
        // *-------------------------------------*
        if (User_Number != 6 )
              Error_Exit( 'Bad call to subroutine
                    Open_File().\n',NULL);
        if (REC_Records != 0 ) {
              // *-------------------------------*
              // |    Wrap the minor recovery number      |
              // |    if needed.                          |
              // *-------------------------------*
              if (REC_Sub_Code == 99 )
                    REC_Sub_Code = 0;
              /**-------------------------------**/
              /*|   Create the recovery file name.       |*/
              /**-------------------------------**/
              fclose( Output_File );
              sprintf( Output_File_Name,'%sREC%02d-
                    %02d.REC', Output_Path,
                    ++REC_Sub_Code, REC_Code );
              Bytes_Free -= Cluster_Size;
              Display_Filename( Output_File_Name );
              Records = 0;
              /**-------------------------------**/
              /*|   Open the recovery file for output.   |*/
              /**-------------------------------**/
              if ( ( Output_File = fopen( Output_File_Name,
                    'w' ) ) == NULL )
              Error_Exit('I can not open the fil*@&g
                    recovery file (%s)
                    !!!\n',Output_File_Name);
              REC_Records = 0;
        }
        return;
}
else
        {
        Data = Output_Buffer + Offset;
        if ( strlen( Data ) == 6 ) {
              // *-------------------------------*
              // | Oops!  We seem to have slipped on a.  |
              // | bug.  Try to get data from next line  |
              // *-------------------------------*
              // Point to next line
              Data = Data + 7;
              // Search for first comma in line.
              Pointer = strchr( Data, ',' );
              if ( Pointer - Data > 2 || Pointer - Data < 1 ) {
                    // *-------------------------*
                    // |          Death by Bunga!!!!     |
                    // *-------------------------*
                     strcpy( U_Number,'13');
                  }
              else
                    strncpy(U_Number,Data,Pointer-Data );
              }
          else
              strncpy( U_Number, Data, 2 );
        User_Number = atoi( U_Number );
        if (User_Number <1 || User_Number > 10 )
              User_Number = 13;
        fclose( Output_File );
        if ( User_Number == 6 )       {
              if (REC_Code == 99 )
                 REC_Code = 0;
              sprintf( Output_File_Name,'%sREC%02d-
                 %02d.REC', Output_Path, 1,
                 ++REC_Code );
              REC_Sub_Code = 1;
           }
         else
           {
              if (Use_Code[User_Number] == 99 )
                 Use_Code[User_Number] = 0;
              sprintf( Output_File_Name,'%sUNK%1d-
                    %02d.DTE', Output_Path,
                    User_Number, ++Use_Code[
                    User_Number] );
           }
Records = 0;
if (User_Number != 6 ) {
              Bytes_Free -= Cluster_Size;
              Display_Filename( Output_File_Name );
              if ( (Output_File = fopen( Output_File_Name,
                    'w' ) ) == NULL )
                    Error_Exit('I can not open the fil*@&g
                          date file (%s) !!!\n',
                          Output_File_Name);
              // Strip off user number
              strtok( Data,',' );
              // Strip off User number description
              Wd = strtok( NULL, ',' );
              strcpy( Assay_Name, Wd );
              if ( Wd == NULL )
                    strcpy( Assay_Name, 'ERROR' );
              for (i = 0; i < strlen(Assay_Name); i++ ) {
                    if (Assay_Name[i] == ' ') {
                          Assay_Name[i] = 0;
                          i = 100;
                    }
              }
              strtok( NULL, ',' );       // Strip off counting time
              // Get the day of week abbreviation.
              Wd = strtok( NULL, ',' );
              // Get day of month
              Da = strtok( NULL, ',' );
              if ( Da == NULL )
                    Da = '1' ;
              // Get month abbreviation.
              Mo = strtok( NULL, ',' );
              Yr = strtok( NULL, ',' ); // Get the year
              if ( Yr == NULL )
                    Yr = '1980';
              for (i = 0; i < 7; i++ ) {
                    Day_Code = 0;
                    if ( !strcmp(Wd, Abriev_Weekday[i] ) ) {
                          Day_Code = i;
                          i = 10;
                    }
              }
              for (i = 0; i < 12; i++ ) {
                    Month_Code = 0;
                    if ( !strcmp(Mo, Abriev_Month[i] ) ) {
                          Month_Code = i;
                          i = 15;
                    }
              }
              if ( (fprintf( Output_File,'%s\n%s\n%s\n%s\n',
                    Weekday[Day_Code],
                    Month[Month_Code] , Da, Yr ) ) == EOF )
```

```
                Error_Exit ('Error during creation of datefile
                    (%s)!!\n', Output_File_Name );
                fclose( Output_File );
            }
            if ( User_Number == 6 )
              {
                REC_Records = 0;
              }
            else
                strcpy ( Output_File_Name + strlen(
                    Output_File_Name ) - 3, 'UNK' );
            Records = 0;
            Bytes_Free -= Cluster_Size;
            Display_Filename( Output_File_Name );
            if ( (Output_File = fopen( Output_File_Name, 'w' ) )
                == NULL )
                Error_Exit('I can not open the fil*@&g output
                    file (%s) !!!\n',Output_File_Name);
        }
}


// *-------------------------------*
// | PROCEDURE: Write_Data         |
// |-------------------------------|
// | PURPOSE: Write data to the disk.  |
// | ON CALL: Nothing              |
// | RETURNS: Nothing              |
// *-------------------------------*
void Write_Data() {
    unsigned int Next, Current=0, Data_Offset;
    char *Dose, *CPM, Scratch[20];

    Dose = NULL;
    /* if there really is no data, then return */
    if ( Output_Tail == 0 )
        return;
    if ( strlen(Output_File_Name) != 0 )
        if ( ( Output_File = fopen( Output_File_Name, 'a' ) ) ==
            NULL )
            Error_Exit('I can not open the fil*@&g output file
                (%s) !!!',Output_File_Name);
    while ( TRUE ) {
      /* Scan for end of string */
      if ( (Next = Scan_for_NULL( Current ) ) == 0 ) {
            /**-------------------------**/
            /*| No terminating NULL found |*/
            /**-------------------------**/
            No_NULL_Found( Current );
            return;
      }
      if (strlen( Output_Buffer + Current ) == 6 ) {
            switch ( Get_Code( Current ) ) {
                case 200:       /* Beginning of User Number
      */
                    Data_Offset = Next;
                    if
     ((Next=Scan_for_NULL(Data_Offset))== 0 ) {
                    /* No NULL so not complete line */
                    No_NULL_Found( Current );
                    return;
                }
                Open_File( 0, Data_Offset );
                    Display_Status( 200 );
                    break;
                case 999:       /* End of User Number */
                    fclose( Output_File );
                    Output_File_Name[0] = 0;
                    Records = 0;
                    Display_Status( 999 );
                    break;
                case 52:        /* Replicate Average UNK. */
                    Data_Offset = Next;
                    if ( (Next = Scan_for_NULL (
                        Data_Offset ) ) == 0 ) {
                      /* No NULL so not complete line */
                      No_NULL_Found( Current );
                      return;
                    }
                    strtok( Output_Buffer + Data_Offset, ',' );
                    /* Strip off CV. */
                    strtok( NULL, ',' ); /* Strip off % Bound */
                    if (Dose == NULL )
                        printf('Data_offset=%s',
                            Output_Buffer + Data_Offset);
                    Dose = strtok( NULL, ',' );
                    fprintf( Output_File,'%s\n', Dose );
                    Bytes_Free -= ( strlen( Dose ) + 2 );
                    Display_Num_Records( ++Records );
                    Display_Status( 52 );
                    break;
                case 100:       /* Recovery Count. */
                    Data_Offset = Next;
                    if ((Next=Scan_for_NULL(Data_Offset))
                        == 0 ) {
                        /* No NULL so not complete line */
                        No_NULL_Found( Current );
                        return;
                    }
                    /* Strip out Sample Number */
                    strtok( Output_Buffer + Data_Offset, ',' );
                    /* Strip off Position */
                    strtok( NULL, ',' );
                    /* Strip off First selected channel # */
                    strtok( NULL, ',' );
                    /* Get counts per minute. */
                    CPM = strtok( NULL, ',' );
                    if (atof( CPM ) > 1200.0 ) {
                        Display_Status( 200 );
                        Open_File( 1, 0 );
                    }
                    fprintf(Output_File,'%.2f\n', atof(CPM) );
                    sprintf( Scratch, '%.2f\n', atof( CPM ) );
                    Bytes_Free -= strlen( Scratch );
                    Display_Num_Records( ++Records );
                    ++REC_Records;
                    Display_Status( 100 );
                    break;
                default:
                    Display_Status( 69 );
                    break;
            }
      }
      Current = Next;
    } /* End while loop. */
}

void
No_NULL_Found ( unsigned int Current ) {
    if ( Current != 0 )
        Move_Data_Up( Current );
    Output_Data_Waiting = 0;
    if ( Output_File != NULL ) {
        fclose ( Output_File );
    }
}
```

```
int Get_Code( unsigned int Code_Offset ) {
        int Block_Code, Check_Sum, Calc_Sum=0, i;

        sscanf(Output_Buffer+Code_Offset,"%d,%X", &Block_Code,
                &Check_Sum );
        for (i=0; i<4; i++ )
                Calc_Sum += Output_Buffer[Code_Offset+i];
        if ( Calc_Sum != Check_Sum )
                Check_Sum_Error++;
        return ( Block_Code );
}


void
Move_Data_Up ( unsigned int Start ) {
    int i;
    int j = 0;

    /**-------------------------------------------**/
    /*| Move the incomplete data to beginning of the buffer |*/
    /**-------------------------------------------**/
    for ( i=Start; i < Output_Tail; i ++ )
            Output_Buffer[j++] = Output_Buffer[i];
    /**-------------------------------------------**/
    /*| Reset the Output Buffer tail pointer.      |*/
    /**-------------------------------------------**/
    Output_Tail = j;
}

/*
 *-------------------------------------------------------*
 | Procedure: Reset_Timer                                |
 *-------------------------------------------------------*
 | Resets the Last_Time variable                         |
 *-------------------------------------------------------*
 | Parameters: (none)                                    |
 | Returned: (none)                                      |
 | Variables: time_t Last_Time          Global variable
containing  |
 |                                  last time routine called.  |
 *-------------------------------------------------------*
 */
void
Reset_Timer() {
        Last_Time = 0;
}

// *-------------------------------------------------------*
// | Procedure: Elapsed_Time                               |
// |-------------------------------------------------------|
// | Determines how many seconds have elapsed since last call. |
// |-------------------------------------------------------|
// | Parameters: (none)                                    |
// | Returned: Elapsed time in seconds since last call.    |
// | Variables: time_t Current_Time Current time in seconds. |
// |            time_t E_Time       Elapsed time.          |
// |            time_t Last_Time    Global variable containing |
// |                                last time routine called.  |
// *-------------------------------------------------------*
time_t
Elapsed_Time() {
        time_t Current_Time, E_Time;

        Current_Time = time( NULL );
        if (Last_Time == 0 )
                {
                    Last_Time = Current_Time;
                    return (0);
                }
            else
                {
                    E_Time = Current_Time - Last_Time;
                    return ( E_Time );
                }
}

void
Initialize_Video( void ) {
        draw_box(1, 1, 80, 25);
        gotoxy( 33, 1 );
        cputs("LSCdata (1.2)");
        draw_box( 3, 3, 23, 23 );
        gotoxy( 5, 3 );
        cputs("Filename");
        gotoxy( 16, 3 );
        cputs("Records");
        gotoxy( 53,2 );
        cprintf("Bytes Received:%10u",0 );
        draw_box( 58, 19, 78, 24);
        gotoxy( 60, 20 );
        cputs("Approximate Bytes");
        gotoxy( 62, 21 );
        cputs("Free on Disk");
        draw_box(32, 15, 78, 17);
        gotoxy( 50, 15 );
        cputs("Status");
        Display_Status( 0 );
        gotoxy(6,25);
        cputs("Press the F10 key when ready to remove the disk");
}

int
Control_Break( ) {
        /* a noise, then return 0.          */
        sound(110);
        delay(200);
        nosound();
        return ( TRUE );
}


void
Capture_Interrupts() {
        _harderr( Hardware_Error );
        ctrlbrk( Control_Break );
        Enable_INT_Handler();
}



// *-------------------------------------------------------*
// | Procedure: Allocate_Buffers                           |
// |-------------------------------------------------------|
// | Allocates serial input and output buffers             |
// |-------------------------------------------------------|
// | Parameters: (none)                                    |
// | Returned:    (void)                                   |
// | Variables: char *COM_buffer    Global pointer to the  |
// |                                allocated input buffer.|
// |            char *Output_Buffer Global pointer to the  |
// |                                allocated output buffer.|
// *-------------------------------------------------------*
void
Allocate_Buffers() {
```

```
        if ((COM_Buffer = (char *)malloc(BUFFER_SIZE)) ==
                NULL) {
                fprintf(stderr,"Fatal error: Unable to allocate serial
                communications buffer.\n");
                exit(100);
        }
        if ((Output_Buffer = (char *)malloc(BUFFER_SIZE)) ==
                NULL) {
                fprintf(stderr,"Fatal error: Unable to allocate output
                buffer.\n");
                exit(100);
        }
}

void promptForNewDisk() {
        struct ffblk File_Block;
        FILE *dummy;
        char box_save[( BOX_LR_X-BOX_UL_X + 2) * (BOX_LR_Y
                - BOX_UL_Y +2 ) * 2 ];
        char Search_String[20];

        Bytes_Received=0;
        Window_Line=1;
        Initialize_Video();
        // *-------------------------------------------*
        // |  Set global varible to tell error handling routine  |
        // |  that we will want to prompt for a disk if none     |
        // |  is in the drive.                                   |
        // *-------------------------------------------*
        #pragma warn -aus  // Stop annoying warning
        Prompt_Disk = 1;
        #pragma warn +aus
        // *-------------------------------------------*
        // | Check for disk in drive, and if not there.  |
        // | force an error so we can get the user to    |
        // | Put it in.                                  |
        // *-------------------------------------------*
        strcpy(Search_String, Disk_Drive);
        strcat( Search_String, "lscdsk.tst");
        while ((dummy=fopen ( Search_String,"w" )) == NULL )
                ;       /*Loop until we found disk */
        fclose (dummy);
        unlink( Search_String );
        strcpy(Search_String, Disk_Drive);
        strcat( Search_String, "lscdata.exe");
        while ( !findfirst( Search_String , &File_Block, 0 )) {
                gettext( BOX_UL_X, BOX_UL_Y, BOX_LR_X,
                        BOX_LR_Y+1, box_save );
                highvideo();
                draw_box( BOX_UL_X, BOX_UL_Y, BOX_LR_X,
                        BOX_LR_Y+1 );
                gotoxy( BOX_UL_X+3, BOX_UL_Y+1 );
                _setcursortype(_NOCURSOR);
                cputs("      This is the LSC data acquisition program
                        disk.");
                gotoxy( BOX_UL_X+3, BOX_UL_Y + 2 );
                cputs( "Please insert a blank formated disk, then
                        push the ENTER key." );
                while ( getch()     != 13 ) {
                        sound(110);
                        delay(200);
                        nosound();
                }
                puttext( BOX_UL_X, BOX_UL_Y, BOX_LR_X,
                        BOX_LR_Y+1, box_save );
                _setcursortype(_NORMALCURSOR);
                lowvideo();
                _dos_getdiskfree( Drive, &Disk_Data);
                biosdisk( RESET_DRIVE, 0, 0, 0, 0, 0, NULL);
        }
        strcpy(Search_String, Disk_Drive);
        strcat( Search_String, "*.*");
        while ( !findfirst( Search_String, &File_Block, 0 )) {
                biosdisk( RESET_DRIVE, 0, 0, 0, 0, 0, NULL);
                gettext( BOX_UL_X, BOX_UL_Y, BOX_LR_X,
                        BOX_LR_Y+1, box_save );
                highvideo();
                draw_box( BOX_UL_X, BOX_UL_Y, BOX_LR_X,
                        BOX_LR_Y+1 );
                gotoxy( BOX_UL_X+3, BOX_UL_Y+1 );
                _setcursortype(_NOCURSOR);
                cputs("                This is not a blank disk.
                                ");
                gotoxy( BOX_UL_X+3, BOX_UL_Y + 2 );
                cputs( "Please insert a blank formated disk, then
                        push the ENTER key." );
                while ( getch()     != 13 ) {
                        sound(110);
                        delay(200);
                        nosound();
                }
                puttext( BOX_UL_X, BOX_UL_Y, BOX_LR_X,
                        BOX_LR_Y+1, box_save );
                _setcursortype(_NORMALCURSOR);
                lowvideo();
                biosdisk( RESET_DRIVE, 0, 0, 0, 0, 0, NULL);
                flushall();
                _dos_getdiskfree( Drive, &Disk_Data);
                Display_Free();
        }
        _dos_getdiskfree( Drive, &Disk_Data );
        Bytes_Free = ( unsigned long int )Disk_Data.avail_clusters
                * ( unsigned long int )Disk_Data.bytes_per_sector *
                ( unsigned long int )Disk_Data.sectors_per_cluster;
        Cluster_Size = Disk_Data.sectors_per_cluster *
                Disk_Data.bytes_per_sector;
        #pragma warn -aus // Stop annoying warning
        Prompt_Disk = 0;
        #pragma warn +aus
}
```

3. *Serial Port Interrupt Handler for LSCDATA.EXE*

```
; *-------------------------------------------*
; | SOURCE FILE: COMHNDLR.ASM                 |
; |-------------------------------------------|
; | Serial port data collection routine for use by |
; | LSCdata.exe.                              |
; *-------------------------------------------*
        Title COMHNDLR.ASM  Serial port interrupt handler routines.
        Page 55,132
        ;
```

```
                ; Assembler routines for use with C program.
                ;
                .MODEL small
Comm_Port       EQU     0               ; Set = 0 for COM1, 1 for COM2
BUFFER_SIZE     EQU     10240
;* ------------------------------------ *
;|   Masks for COM Port initialization.  |
;* ------------------------------------ *
P_NONE          EQU     0
P_ODD           EQU     00001000B
P_EVEN          EQU     00011000B
STOP1           EQU     00000000B
STOP2           EQU     00000100B
WORD7           EQU     00000010B
WORD8           EQU     00000011B
BAUD0110        EQU     0
BAUD0150        EQU     00100000B
BAUD0300        EQU     01000000B
BAUD0600        EQU     01100000B
BAUD1200        EQU     10000000B
BAUD2400        EQU     10100000B
BAUD4800        EQU     11000000B
BAUD9600        EQU     11100000B

Pic_Mask        EQU     21H             ; Port adress of 8259 mask register.
Pic_Eoi         EQU     20H             ; Port adress of 8259 EOI instr.
LCR_DLAB        EQU     10000000B

        If      Comm_Port               ; Define physical port assignments.
Comm_Data       EQU     02F8h           ; for COM2.
Comm_Ier        EQU     02F9h           ;
Comm_Iir        EQU     02FAH           ; Interrupt Identification Register.
Comm_Lcr        EQU     02FBH
Comm_Mcr        EQU     02Fch           ;
Comm_Stat       EQU     02Fdh           ;
Com_Int         EQU     0Bh             ;
Int_Mask        EQU     08H             ; Mask for 8529, COM2 is IRQ3

        Else
Comm_Data       EQU     03F8h           ; Define physical port assignments
Comm_Ier        EQU     03F9h           ; for COM1.
Comm_IIR        EQU     03FAH           ; Interrupt Identification Register.
Comm_LCR        EQU     03FBH
Comm_Mcr        EQU     03Fch           ;
Comm_Stat       EQU     03Fdh           ;
Com_Int         EQU     0Ch             ;
Int_Mask        EQU     10H             ; Mask for 8259, COM1 is IRQ4.
        Endif

        .DATA
        EXTRN _COM_Tail:WORD
        EXTRN _COM_Head:WORD
        EXTRN _COM_Buffer:WORD
        EXTRN _COM_Errors:WORD
        EXTRN _Prompt_Disk:BYTE
        EXTRN _COM_Data_Pending:BYTE
Intc_Seg        DW  0
Intc_Offs       DW  0
Init_Msg        DB  'Please place a blank formatted DS/DD disk into the drive.'
Len_Init_Msg    EQU $ - Init_Msg
Init_Msg_2      DB  'Press any key to continue.'
Len_Init_Msg_2  EQU $ - Init_Msg_2
Abort_Msg       DB  'Correct error then press a key: A)bort, R)etry, I)gnore, or F)ail.'
Len_Abort_Msg   EQU $ - Abort_Msg
Video_Segment   DW  0
Vid_Offset      DW  0
Cursor_Type     DW  0
Cursor_Position DW  0
```

```
Active_Page     DB   0
Save_Buffer     DB   64 DUP ('SCREENBUFF')

                .CODE
                EXTRN _Display_Critical_Error
                ASSUME CS:@code, DS:@data

Sound           PROC
                PUSH  AX
                PUSH  BX
                PUSH  CX
                PUSH  DX
                PUSH  DI

                MOV   AL,0B6H
                OUT   43H,AL
                MOV   DX,14H
                MOV   AX,4F38H
                DIV   DI
                OUT   42H,AL
                MOV   AL,AH
                OUT   42H,AL
                IN    AL,61H
                MOV   AH,AL
                OR    AL,3
                OUT   61H,AL
Wait_Here:      MOV   CX,2801
Spkr_On:        LOOP  Spkr_On
                DEC   BX
                JNZ   Wait_Here
                MOV   AL,AH
                OUT   61H,AL

                POP   DI
                POP   DX
                POP   CX
                POP   BX
                POP   AX
                RET

Sound           ENDP

;* ---------------------------------------------------------- *
;| PROCEDURE : Disable_INT_Handler                            |
;|                                                            |
;| Disables the Asyncronous Communications Routine.           |
;|                                                            |
;* ---------------------------------------------------------- *
                PUBLIC _Disable_INT_Handler
_Disable_INT_Handler PROC  NEAR

                PUSH    AX
                PUSH    DX
                PUSH    DS
                MOV     AX,@data            ; Set DS to the data segment.
                MOV     DS,AX               ;
                MOV     AX,Intc_Seg         ;
                ADD     AX,Intc_Offs        ;
                OR      AX,AX               ; Have we taken over the COMM int handler?
                JZ      Disable_INT_Handler_1 ;  If no, then don't try to restore it.
                IN      AL,Pic_Mask         ; Read current 8259A int mask.
                OR      AL,Int_Mask         ; Set mask for this COM port.
                OUT     Pic_Mask,AL         ; Write int mask back to 8259A.

                MOV     DX,Intc_Offs        ; Get the saved INT handler offset.
                MOV     AX,Intc_Seg         ; Get the saved INT handler segment.
                MOV     DS,AX               ; Make DS point to the INT handler segment.
                MOV     AH,25H              ; Load DOS function 'Set Vector'.
```

```
                MOV     AL,Com_Int          ; Load INT number for COM port.
                INT     21H                 ; Reset INT handler.
Disable_INT_Handler_1:
                POP     DS
                POP     DX
                POP     AX
                RET

_Disable_INT_Handler  ENDP                  ;

Get_Video_Mode EQU 0FH
Get_Cursor     EQU 03H
IGNORE         EQU 0
RETRY          EQU 1
ABORT          EQU 2
FAIL           EQU 3

;* ------------------------------------------------------ *
;| Procedure : COM_INT_Handler                            |
;|                                                        |
;| Asynchronous Communications Interrupt routine.         |
;| (@l)                                                   |
;* ------------------------------------------------------ *
COM_INT_Handler PROC  FAR
        STI                         ; Enable interrupts.
        PUSH    AX                  ; Save effected registers.
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    DS
        PUSH    ES
        PUSH    DI

        MOV     AX,@data            ; Set the Data segment to the data segment.
        MOV     DS,AX               ;
        MOV     ES,AX
        ;* ------------------------------------------------ *
        ;| Interrupt Bits 2,1                               |
        ;|   1 1  RX Error condition (Highest Priority)     |
        ;|   1 0  RX Character available.                   |
        ;|   0 1  TXD Register empty.                       |
        ;|   0 1  Modem Interrupt (Lowest Priority.)        |
        ;* ------------------------------------------------ *
        MOV     DX,Comm_IIR                 ; Load the Interrupt Identification Register.
        IN      AL,DX                       ; Get the interrupt identification form the UART.
        AND     AL,00000110B                ; Was the interrupt caused by an error?
        CMP     AL,00000110B
        JNE     COM_INT_Handler_1           ;           If no, then continue.
        XOR     AH,AH                       ; Clear high order byte.
        MOV     _COM_Errors,AX              ; Store COM error.
        JMP     SHORT COM_INT_Handler_3     ; End Interrupt.
COM_INT_Handler_1:
        MOV     DX,Comm_Data                ; Load COM port into DX
        MOV     DI,_COM_Tail                 Load the Data Buffer Offset into DI.
        ADD     DI,_COM_Buffer
        CLC                                 ; Clear the Carry Flag.
        CLI                                 ; Clear Interrupts.
        IN      AL,DX                       ; Get a character from the COM port.
        MOV     [DI],AL                     ; Store it in the data buffer.
        CMP     DI,BUFFER_SIZE - 1          ; Was this the last character in th buffer?
        JNE     COM_INT_Handler_2           ; If no, then continue.
        ;
        ;       We are at the end of the COMM Buffer, so we need to
        ;       wrap back to the beginning.
        ;
        MOV     _COM_Tail,0                 ; Wrap the COMM buffer pointer.
        JMP     SHORT COM_INT_Handler_2a ; Jump over other tail calcs.
COM_INT_Handler_2:
```

```
        INC     _COM_Tail                       ; Increment our data pointer to the next location.
COM_INT_Handler_2a:
        OR      _COM_Data_Pending,1             ; Signal that there is data waiting.
COM_INT_Handler_3:
        STI                                     ; Enable interrupts.
        MOV     AL,20H                          ; Signal end of interrupt to controller.
        OUT     Pic_Eoi,AL

        POP     DI
        POP     ES
        POP     DS                              ; Restore registers.
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        IRET

COM_INT_Handler ENDP

;*-------------------------------------------------*
;| PROCEDURE : Enable_INT_Handler                  |
;|                                                 |
;| Enables the Asyncronous Communications Routine. |
;|                                                 |
;*-------------------------------------------------*
        PUBLIC _Enable_INT_Handler
_Enable_INT_Handler PROC  NEAR

        PUSH    AX
        PUSH    BX
        PUSH    DX
        PUSH    DS
        PUSH    ES

        MOV     AH,35H                          ; Load DOS function 'Get Vector'.
        MOV     AL,Com_Int                      ; Load interrupt number for Com port.
        INT     21H                             ; Get the vector.
        MOV     Intc_Seg,ES                     ; Save the Vector segment.
        MOV     Intc_Offs,BX                    ; Save the Vector offset
        PUSH    CS                              ; Make DS point to our code segment.
        POP     DS
        MOV     DX,OFFSET COM_INT_Handler       ; Load Offset of COM_INT Handler.
        MOV     AH,25H                          ; Load DOS function 'Set Vector'.
        MOV     AL,Com_Int                      ; Load interrupt number for COM port.
        INT     21H                             ; Take over the COM interrupt.
        ;
        ;       Set baud rate
        ;
        MOV     AH,00H
        MOV     AL, BAUD0300 OR STOP1 OR P_EVEN OR WORD8
        MOV     DX, Comm_Port
        INT     14H

        MOV     DX,Comm_Mcr                     ; Modem controller DTR and OUT2;
        MOV     AL,0Bh                          ;
        OUT     DX,AL                           ;

        MOV     DX,Comm_Ier                     ; Interrupt enable register
        MOV     AL,1                            ; on asynch controller.
        OUT     DX,AL

        IN      AL,Pic_Mask                     ; Read current 8259 int. mask.
        AND     AL,NOT Int_Mask                 ; Reset mask for this COM port.
        OUT     Pic_Mask,AL                     ; Write back 8259 int. mask.

        POP     ES                              ; Restore used registers.
        POP     DS
        POP     DX
```

```
        POP     BX
        POP     AX

        RET
_Enable_INT_Handler ENDP

        END
```

G. RIA data file copying Module

1. Header File for RIACOPY.C

```
#ifndef _RIACOPY_H_
#define _RIACOPY_H_
// *-------------------------------------*
// | SOURCE FILE: RIACOPY.H              |
// *-------------------------------------*
// | Header file for RIACOPY.C - Copy RIA files. |
// |                                     |
// | RIA program version 1.1             |
// *-------------------------------------*
#define RIGHT       77
#define LEFT        75
#define DELETE      83
#define HOME        71
#define END         79
#define CR          13
#define LF          10


struct Window_Style Main_Window = {RED,  CYAN, RED, 1,
     1, 80, 25, BLACK,  CYAN, 'RIAcopy (1.0)- Copies RIA files
     from floppy' };
struct Window_Style File_Window = {WHITE,  BLUE, WHITE,
     2,  2, 17, 24, WHITE,  LIGHTGRAY, 'Copied Files' };
struct Window_Style Records_Window = {WHITE,  BLUE,
     WHITE,  18,  2, 28, 24, WHITE,  LIGHTGRAY, 'Records'};
struct Window_Style Desc_Window = {WHITE,  BLUE,
     WHITE,  29,  2, 70, 24, WHITE,  LIGHTGRAY,
     'Description' };
struct Window_Style Data_Window = {YELLOW, LIGHTGRAY,
     YELLOW, 67,  2, 79, 13, WHITE,  RED , 'Data' };
struct Window_Style Name_Window = {WHITE,  CYAN,
     YELLOW, 16, 10, 62, 12, WHITE,  CYAN, 'Enter Your Last
     Name [ Max 12 character ]' };
struct Window_Style Get_Descript = {WHITE,  CYAN,
     YELLOW, 18, 20, 63, 22, WHITE,  CYAN, " };

void Delete_File( FILE *, char * );
void Describe( char *, int );
void Print_Stats( int , char * );

/* *-------------------------------* */
/* |        Global Constants       | */
/* *-------------------------------* */
#define TRUE        -1
#define FALSE       0
#define INPUT_UL_X  4
#define INPUT_UL_Y  4
#define INPUT_LR_X  22
#define INPUT_LR_Y  22
#define IGNORE      0
#define RETRY       1
#define ABORT       2
/* *-------------------------------* */
/* |  Data from Prompt for Disk    | */
/* *-------------------------------* */
#define RESET_DRIVE         0
#define VERIFY_SECTORS      4
#define HEAD        0
#define DISK        0
#define TRACK       0
#define SECTOR      1
#define NUMBER_SECTORS      1
#define BOX_UL_X    6
#define BOX_UL_Y    11
#define BOX_LR_X    73
#define BOX_LR_Y    13
#endif
```

2. Source for RIA file copying program

```
// *-------------------------------------*
// | SOURCE FILE: RIACOPY.C              |
// *-------------------------------------*
// | Copies files from the RIA data floppy disks |
// *-------------------------------------*
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>
#include <bios.h>
#include <ctype.h>
#include <dir.h>
#include <process.h>
#include <alloc.h>

#include 'rialib.h'
#include 'riacopy.h'
// *-------------------------------------------------------*
// |          Program - RIAcopy.Exe ( Version 1.1 )        |
// |-------------------------------------------------------|
// | Copies raw data files from the LSC RIA data disks onto the |
// | hard disk and then prompts the user to create 4DOS    |
// | compatible desciptions for the files.                 |
// |                                                       |
// | Version 1.1 - Delete zero length files from RIA data diskettes. |
// *-------------------------------------------------------*


char datapath[120]='C:\\LOTUS\\LSC\\';
char *Time_Stamp='C:\\LOTUS\\RIADATA\\timestam.p';
char User_Name[13]='';
FILE *Stats_File;
struct ffblk ffblk;
int start = 1,
Copied_Files=0;
char Error_Msg[80];
char Prompt_Disk=0;
char Show_Data[10][12];
char *file_desc[150];
char *new_desc[150];
int  fcount=-1;
FILE *Desc_File;
```

```
struct Window_Style *ScreenWindows[] = { &Main_Window,
        &File_Window, &Desc_Window, &Records_Window,
        NULL };


#ifdef DEBUG
        char *Drive='A:';
        int _drive=0;
#else
        char *Drive='B:';
        int _drive=1;
#endif

int
main (int argc /*, char *argv[]*/ ) {
        FILE *current_file;
        char *tmp_desc;
        char Filename[80];
        int  Records;

        if ((tmp_desc = getenv('RIADRIVE')) != NULL ) {
        Drive[0] = tmp_desc[0];
        Drive[1] = ':';
        Drive[2] = 0;
        _drive = toupper( tmp_desc[0] ) - 'A';
        }
        if ((tmp_desc = getenv('RIAPATH')) != NULL )
                strcpy( datapath, tmp_desc );
        if (argc > 1 ) {
                printf('RiaCopy.Exe - Moves RIA files from a floppy
                        disk to the\n');
                printf('                LSC directory on the hard
                        disk.\n\n');
                printf('   It will copy from the B: drive by default, but
                        this can be \n');
                printf('   overridden by setting the environment
                        variable RIADRIVE to a \n');
                printf('   drive.  It will store the files on the default
                        drive, in the \n');
                printf('   directory \\LOTUS\\LSC, unless the
                        environment variable RIAPATH\n');
                printf('   is set.\n');
                exit(0);
        }
        if ( ( Stats_File = fopen( 'C:\\System\\RiaStats.Dat','w') ) ==
                NULL ) {
                fprintf( stderr, 'Can\'t open
                        C:\\System\\RiaStats.Dat\n');
                exit(1);
        }
        setcbrk( 1 );
        Read_Descriptions( datapath, &fcount, file_desc );
        Screen_Setup( ScreenWindows );
        _harderr( Hardware_Error );
        Prompt_New_Disk( &Prompt_Disk, Drive );
        Get_User_Name( &Name_Window, User_Name,
                Time_Stamp );
        while ( ( current_file = Get_File( Filename, Drive, &start ) )
                != NULL ) {
                if ( (Records = Copy_File( current_file, Filename,
                        datapath, &Copied_Files,
                        &File_Window, &Records_Window,
                        Show_Data ) ) != 0 )
                {
                        Delete_File( current_file, Filename );
                        if ( Records > 0 ) {
                                Describe( Filename, Records );
                                Print_Stats( Records, Filename );
                        }
                }
                fclose(current_file);
        }
        Save_Desc( datapath, file_desc, new_desc, Copied_Files,
                fcount );
        //  Save_Desc();
        fclose( Stats_File );
        window(1,1,80,25);
        _wscroll=1;
        clrscr();
        return(0);
}



void
Delete_File( FILE *File, char *Filename ) {
        char *Name;
        Name = allocate( 120 );
        fclose( File );
        strcpy( Name, Drive );
        strcat( Name, Filename );
        unlink( Name );
        free( Name );
}


#pragma warn -par
char *
Copy_Desc( char *Desc ) {
        char *tmp;
        tmp= strdup( Desc );
        if ( tmp == NULL )
                die( 'Can\'t allocate string space for description\n');
        return (tmp);
}

int
Date_File( char * filename ) {
        char * pointer;

        pointer = strchr( filename, '.');
        if (pointer == NULL ) return (0);
        ++pointer;
        if ( strncmpi( pointer, 'dte', 3 ) )
                return(0);
          else
                return(-1);
}

void
Describe( char *Filename, int Records ) {
        int i, length, Line;
        struct text_info ti;
        char msg[80];
        char Description[41]='';
        char *savedesc, *savedata;
        /*  char delete[3]={32,8,0};
        char backspace[5]={8,32,8,0};
        */
        gettextinfo( &ti );
        savedata = Window_Save(&Data_Window);
        savedesc = Window_Save( &Get_Descript );
        if ( Date_File( Filename ) )
                strcpy( Description, 'Date file.' );
        else
```

```
        {
        Window_Box(&Data_Window);
        window(Data_Window.x1+1, Data_Window.y1+1,
        Data_Window.x2-1, Data_Window.y2-1);
        _wscroll=0;
        textcolor( Data_Window.textcolor);
        textbackground( Data_Window.backcolor);

        length = ( Records < 10 ) ? Records: 10 ;
        for ( i=0; i<=length; i++ ) {
              gotoxy(1,i);
              cprintf("%10s",&Show_Data[i-1][0]);
        }
        _wscroll=1;
        window(1,1,80,25);
        textcolor( Get_Descript.textcolor);
        textbackground( Get_Descript.backcolor);
        sprintf( msg, "Describe %s [40 characters max.]",
        Filename );
        strcpy( Get_Descript.title,msg);
        Window_Box( &Get_Descript );
        strcpy(Description, User_Name );
        strcat(Description,": ");
        gotoxy(Get_Descript.x1+1, Get_Descript.y1+1);
        cputs(Description);
        length = 40 - strlen( Description );
        while ( strlen( Description ) == 40 - length ) {
              Input_String( Description + strlen( Description
        ), length, 0, 0 );
              if ( strlen( Description ) == 40 - length ) {
                    Beep();
              }
        }
        Window_Restore( &Get_Descript, savedesc );
        Window_Restore( &Data_Window, savedata );
        }
    window( Desc_Window.x1+1, Desc_Window.y1+1,
        Desc_Window.x2-1, Desc_Window.y2-1);
    textcolor( Desc_Window.textcolor);
    textbackground( Desc_Window.backcolor);
    Line = Copied_Files;
    if (Copied_Files > (Desc_Window.y2 - Desc_Window.y1-1) ) {
        Line = Desc_Window.y2 - Desc_Window.y1 - 1;
        gotoxy( 1,Line );
        cputs("\r\n");
    }
    gotoxy( 1, Line );
    cprintf( "%s",Description );
    window( 1,1,80,24);
    textattr( ti.attribute );
    strcpy( msg, Filename );
    strcat( msg, " " );
    strcat( msg, Description );
    new_desc[Copied_Files-1] = Copy_Desc( msg );
}

void
Print_Stats( int Records, char *Filename ) {
    char *tmp;
    char msg[50];
    if ((tmp = strchr( new_desc[Copied_Files-1], ' '))==NULL ) {
        sprintf(msg,"Bad description for %s.\n",Filename );
        die( msg );
    }
    ++tmp;
    fprintf(stdprn,"%-13s %-40s \n Contains %d records\n",
        Filename, tmp, Records );
}
```

H. RIA File description Module - RIADESC.EXE

1. Header File For RIADESC.C

```
#ifndef _RIADESC_H_
#define _RIADESC_H_
// *-------------------------------------------*
// | SOURCE FILE: RIADESC.H                    |
// |-------------------------------------------|
// | Header File for RIADESC.C - Describe RIA files.  |
// |                                           |
// | RIA program version 1.1                   |
// *-------------------------------------------*
struct Window_Style Main_Window = {RED, CYAN, RED,1,
    1, 80, 25, BLACK, CYAN, "RIAdesc (1.01) - Describe RIA
    worksheets" };
struct Window_Style File_Window = {WHITE, BLUE, WHITE,
    10, 5, 70, 20, WHITE, LIGHTGRAY, "File Actions" };
struct Window_Style Get_Descript = {WHITE, BROWN,
    YELLOW, 2, 20, 78, 23, WHITE, BROWN, " };
struct Window_Style Name_Window = {WHITE, CYAN,
    YELLOW, 16, 10, 62, 12, WHITE, CYAN, "Enter Your Last
    Name [ Max 12 character ]" };

int Get_Time_Stamp( void );
time_t relative_time( struct ffblk );

GetFileToDescribe( struct ffblk *fileBlock );
char * get_123_title( char * );
void Delete_File( FILE *, char * );
void Describe( void );
void Print_Stats( int , char * );
/* *-------------------------------------* */
/* |          Global Constants           | */
/* *-------------------------------------* */
#define TRUE          -1
#define FALSE         0
#define INPUT_UL_X    4
#define INPUT_UL_Y    4
#define INPUT_LR_X    22
#define INPUT_LR_Y    22
#define IGNORE        0
#define RETRY         1
#define ABORT         2
/* *-------------------------------------* */
/* | Data from Prompt for Disk           | */
/* *-------------------------------------* */
#define RESET_DRIVE       0
#define VERIFY_SECTORS    4
#define HEAD              0
#define DISK              0
#define TRACK             0
#define SECTOR            1
#define NUMBER_SECTORS    1
#define BOX_UL_X          6
#define BOX_UL_Y          11
#define BOX_LR_X          73
#define BOX_LR_Y          13
#endif
```

2. Source for RIADESC.C

```
// *-------------------------------------------*
// | SOURCE FILE: RIADESC.C                    |
```

```
// |-------------------------------------|
// | Assigns descriptions to RIA files.  |
// *-------------------------------------*
/* #define TESTING /* */
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>
#include <bios.h>
#include <ctype.h>
#include <dir.h>
#include <process.h>
#include <alloc.h>
#include <time.h>

#include "rialib.h"
#include "riadesc.h"
// *-------------------------------------------------------*
// |          Program - RIAdesc.Exe ( Version 1.1 )        |
// |-------------------------------------------------------|
// |  Has the user create 4DOS compatible desciptions for  |
// | newly created RIA 1-2-3 worksheet files.              |
// |                                                       |
// |  Version 1.1 -   Make sure we don't try to describe   |
// |                  worksheets that are already described.|
// *-------------------------------------------------------*

struct Window_Style *screenWindows[]={ &Main_Window,
&File_Window, NULL };
char datapath[120]="C:\\LOTUS\\RIADATA\\";
char User_Name[13]="";
FILE *Stats_File;
struct ffblk ffblk;
char *searchname;
int start  = 1,
F_Line = 1,
Copied_Files=0;
char Error_Msg[80];
char Prompt_Disk=0;
char Show_Data[10][12];
char *file_desc[500];
char *new_desc[100];
time_t Time_Stamp;
int  fcount=-1;
FILE *Desc_File;
int _Describe=0;
int _First=1;

#ifdef TESTING
#     define DEBUGWAIT( A ) puts( A) ; getch()
#  else
#     define DEBUGWAIT( A )
#endif

#ifdef DEBUG
      char *Drive="A:";
      int  _drive=0;
#else
      char *Drive="B:";
      int  _drive=1:
#endif

int
main (int argc /*, char *argv[]*/ ) {
      char *tmp_desc;

      if ((tmp_desc = getenv("RIADRIVE")) != NULL ) {
            Drive[0] = tmp_desc[0];
            Drive[1] = ':';
            Drive[2] = 0;
            _drive = toupper( tmp_desc[0] ) - 'A';
      }
      if ((tmp_desc = getenv("RIAPATH")) != NULL )
            strcpy( datapath, tmp_desc );
      if (argc > 1 ) {
            printf("RiaDesc.Exe - Moves RIA files from a floppy
                  disk to the\n");
            printf("              LSC directory on the hard
                  disk.\n\n");
            printf("  It will copy from the B: drive by default, but
                  this can be \n");
            printf("  overridden by setting the environment
                  variable RIADRIVE to a \n");
            printf("  drive.  It will store the files on the default
                  drive, in the \n");
            printf("  directory \\LOTUS\\LSC, unless the
                  environment variable RIAPATH\n");
            printf("  is set.\n");
            exit(0);
      }
      setcbrk( 1 );
      Read_Descriptions( datapath, &fcount, file_desc );
      Screen_Setup( screenWindows );
      if (!Get_Time_Stamp())
            exit(1);
      searchname = allocate( 150 );
      strcpy( searchname, datapath );
      strcat( searchname, "*.wk1" );
      while ( GetFileToDescribe( &ffblk ) ) {
            Describe( );
      }
      Save_Desc( datapath, file_desc, new_desc, Copied_Files,
            fcount );
      fclose( Stats_File );
      window(1,1,80,25);
      _wscroll=1;
      clrscr();
      return(0);
}

int
GetFileToDescribe( struct ffblk *fileBlock ) {
      time_t file_time;
      int done=0;

      while ( !done ) {
            if ( start ) {
                  done = findfirst( searchname, fileBlock, 0 );
                  start = 0;
               }
            else
               done = findnext( fileBlock );
            if ( done )
                  return( NULL );
            file_time = relative_time( *fileBlock);
            if ( file_time > Time_Stamp ) {
                  return( 1 );
            }
      }
      /* Should never get here */
      return( 0 );
}


int
```

```
Get_Time_Stamp() {
        char *Time_Stamp_Name;


        Time_Stamp_Name = allocate( 140 );
        strcpy( Time_Stamp_Name, datapath );
        strcat( Time_Stamp_Name,'timestam.p');
        if ( findfirst( Time_Stamp_Name, &ffblk, 0 ) != 0 ) {
                free( Time_Stamp_Name );
                return (0);
        }
        Time_Stamp = relative_time( ffblk );
        free( Time_Stamp_Name );
        return ( 1 );
}

time_t
relative_time( struct ffblk file ) {
        struct tm Time;

        Time.tm_sec  = ( file.ff_ftime & 0x1F ) * 2;
        Time.tm_min  = ( file.ff_ftime & 0x7e0 ) >> 5;
        Time.tm_hour = ( file.ff_ftime & 0xF800 ) >> 11;
        Time.tm_mday = ( file.ff_fdate & 0x1F );
        Time.tm_mon  = ( file.ff_fdate & 0x1e0 ) >> 5;
        Time.tm_year = (( file.ff_fdate & 0xfe00 ) >> 9 ) + 80;
        return( mktime(&Time) );
}


char *
Copy_Desc( char *Desc ) {
        char *tmp;
        tmp= strdup( Desc );
        if ( tmp == NULL )
                die( 'Can\'t allocate string space for description\n');
        return (tmp);
}



void
Describe( ) {
        int length, Line, part_len,len,j;
        struct text_info ti;
        char msg[80];
        char Description[41]="";
        char *savedesc, *killfile;
        char *wrksht_title;
        char *wrksht_file;
        int _described_ = FALSE;

        if ( _First ) {
                Get_User_Name( &Name_Window, User_Name,
                        'C:\\LOTUS\\RIADATA\\timestam.p' );
                _First++;
        }
        _Describe=1;
        gettextinfo( &ti );
        /**------------------------------**/
        /*|  See if already described     |*/
        /**------------------------------**/
        len = strlen( ffblk.ff_name );
        for ( j=0; j<fcount; j++ ) {
                if ( strncmpi( ffblk.ff_name, file_desc[j], len ) == 0 ) {
                        _described_ = TRUE;
                        break;
                }
        }
        if ( ! _described_ ) {
                savedesc = Window_Save( &Get_Descript );
                _wscroll=1;
                window(1,1,80,25);
                textcolor( Get_Descript.textcolor);
                textbackground( Get_Descript.backcolor);
                sprintf( msg, 'Describe %s [40 characters max.] Press
                        F2 to delete file.', ffblk.ff_name );
                strcpy( Get_Descript.title,msg);
                Window_Box( &Get_Descript );
                if ((wrksht_file = (char *)malloc ( 128 ))== NULL )
                        {
                                printf( 'Error allocating string space in
                                        procedure DESCRIBE of RIAdesc.c
                                        module.\n');
                                printf( 'Press any key to continue:');
                                getch();
                                wrksht_title = NULL;
                        }
                else
                        {
                                strcpy( wrksht_file, datapath );
                                strcat( wrksht_file, ffblk.ff_name );
                                wrksht_title = get_123_title( wrksht_file );
                        }
                gotoxy(Get_Descript.x1+1, Get_Descript.y1+1);
                cprintf('Worksheet Title: %40.40s',wrksht_title ==
                        NULL ? 'Not titled.' : wrksht_title );
                strcpy(Description, User_Name );
                strcat(Description,': ');
                part_len = strlen( Description );
                length = 40 - part_len;
                if ( wrksht_title != NULL )
                        strncat( Description, wrksht_title, 40-part_len );
                gotoxy(Get_Descript.x1+1, Get_Descript.y1+2);
                textcolor( Get_Descript.forecolor );
                cputs(Description);
                while ( part_len == 40 - length ) {
                        if ((Input_String( Description + 40 - length,
                                length, strlen(Description)-part_len,
                                _Describe )) == INPUT_BAD ) {
                                        _Describe=0;
                                        Window_Restore( &Get_Descript,
                                                savedesc );
                                        window( File_Window.x1+1,
                                        File_Window.y1+1,
                                        File_Window.x2-1,
                                        File_Window.y2-1);
                                        textcolor( File_Window.textcolor);
                                        textbackground(
                                        File_Window.backcolor);
                                        killfile = allocate ( 130 );
                                        strcpy( killfile, datapath );
                                        strcat( killfile, ffblk.ff_name );
                                        unlink( killfile );
                                        Line = F_Line;
                                        if (Copied_Files > (File_Window.y2
                                                - File_Window.y1 - 1 ) )  {
                                                Line=File_Window.y2-
                                                        File_Window.y1-1;
                                                gotoxy( 1,Line );
                                                cputs('\r\n');
                                        }
                                        gotoxy( 1, Line );
                                        cprintf( '%s deleted',killfile );
                                        ++F_Line;
                                        window( 1,1,80,24);
```

```
                    free( killfile );
                    textattr( ti.attribute );
                    return;
                }
            else
                part_len = strlen( Description );
            if ( part_len == 40 - length ) {
                Beep();
            }
        } // End While
        _Describe=0;
        Window_Restore( &Get_Descript, savedesc );
        window( File_Window.x1+1, File_Window.y1+1,
            File_Window.x2-1, File_Window.y2-1);
        textcolor( File_Window.textcolor);
        textbackground( File_Window.backcolor);
        Line = F_Line;
        if (Copied_Files > (File_Window.y2 - File_Window.y1
            - 1 ) )  {
            Line = File_Window.y2 - File_Window.y1 - 1;
            gotoxy( 1,Line );
            cputs("\r\n");
        }
        gotoxy( 1, Line );
        cprintf( "%s described.",ffblk.ff_name );
        ++F_Line;
        window( 1,1,80,24);
        textattr( ti.attribute );
        strcpy( msg, ffblk.ff_name );
        strcat( msg, " " );
        strcat( msg, Description );
        new_desc[Copied_Files++] = Copy_Desc( msg );
    } // End if not described.
} // End  Procedure DescribeDescribe
#pragma warn +par
```

I. RIADIR.EXE - Generates Directory Listings for raw RIA data files.

1. *Header File for RIADIR.C*

```
#ifndef _RIADIR_H_
#define _RIADIR_H_
// *-------------------------------------------*
// | SOURCE FILE: RIADIR.H                     |
// |-------------------------------------------|
// | Header file for RIADIR.C - Get RIA Directory |
// |                                           |
// | RIA program version 1.1                   |
// *-------------------------------------------*
#define RIGHT       77
#define LEFT        75
#define DELETE      83
#define HOME        71
#define END         79
#define CR          13
#define LF          10
#define SPACE       32
#define F10         68
#define UP          72
#define DOWN        80

#define LINE        0
#define ALL         1
#define NONE        2

struct File_Data {
        char *name[150];
        char *desc[150];
        char tag[150];
};

typedef struct File_Stats {
        char *name;
        char *desc;
        char tag;
} File_Stats;


struct Window_Style Main_Window =  {RED,   CYAN, RED,1,
        1, 80, 25, BLACK,  CYAN, 'RIAdir (1.0) - Sorts RIA data
        files for LOTUS.' };
struct Window_Style Info_Window =  {WHITE,  BLUE, WHITE,
        30, 10 , 50, 12, WHITE,  RED       , " };

void Highlight( struct Window_Style );
void Normal( struct Window_Style );
void Print( File_Stats * );
void activate_window( struct Window_Style Window );
void Tag( void );
int get_key( void );
int Scroll_Up (void );
int Scroll_Down (void );
void Update_Window( int );
File_Stats * strucalloc( void );
unsigned long Get_Sort_Code( char * );
int sort_function( const void *, const void * );
char * Find_Desc( char * );
void Delete_File( FILE *, char * ),
void Describe( char *, int );
void Print_Stats( int , char * );
char *getFileName( char *Extension );
#endif
```

2. *Source for RIADIR module*

```
// *-------------------------------------------*
// | SOURCE FILE: RIADIR.C                     |
// |-------------------------------------------|
// | Makes sorted listings of various raw RIA data |
// | files for use by other modules.           |
// *-------------------------------------------*
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>
#include <ctype.h>
#include <dir.h>
#include <process.h>
#include <alloc.h>

#include "rialib.h"
#include "riadir.h"
// *-------------------------------------------*
// |          Program - RIAdir.Exe  (Version 1.0)  |
// |-------------------------------------------|
// | Sorts the directory of the RIA files and creates the files |
// | used by LOTUS to allow the user to select RIA files. |
// |                                           |
// | Compiled using Borland C/C++ 3.1.         |
// *-------------------------------------------*
char datapath[120]="C:\\LOTUS\\LSC\\";
```

```
struct ffblk ffblk;
File_Stats *File_Info[150];
char *extensions[] = { 'unk', 'rec', 'dte' };
int file_count=0;
int start = 1;
char Error_Msg[128];
char *file_desc[150];
int  fcount=-1;
int    Window_Start,
       win_line,
       win_length,
       file_line;
extern struct text_info TEXT;
char Prompt_Disk=0;



int
main (int argc /*, char *argv[]*/ ) {
       char *tmp_desc, outfile[128];
       char *screen;
       int  i, j;
       FILE *output;

       gettextinfo( &TEXT );
       if ((tmp_desc = getenv('RIAPATH')) != NULL )
              strcpy( datapath, tmp_desc );
       if (argc > 1 ) {
              printf('Riadir.Exe -  Sorts RIA data files for use with
                     the LOTUS RIA worksheet.\n\n');
              exit(0);
       }
       setcbrk( 1 );
       Read_Descriptions( datapath, &fcount, file_desc );
       screen = Window_Save( &Main_Window );
       Window_Box( &Main_Window );
       Window_Box( &Info_Window );
       activate_window( Info_Window );
       gotoxy( 4,1);
       cputs('Sorting files');
       File_Info[file_count] = strucalloc();
       for (i = 0; i < 3; i++ ) {
              start = 1;
              while ( ( File_Info[file_count]->name = getFileName(
                     extensions[i] ) ) != NULL ) {
                     File_Info[file_count]->desc = Find_Desc(
                     File_Info[file_count]->name );
                     File_Info[file_count]->tag = 0;
                     ++file_count;
                     File_Info[file_count] = strucalloc();
              }
              strcpy( outfile, datapath );
              strncat( outfile, extensions[i],1);
              strcat(outfile, 'files.dat');
              if ((output = fopen( outfile, 'w' )) == NULL) {
                     sprintf( Error_Msg,'Error opening %s.',outfile );
                     die(Error_Msg);
              }
              if (file_count > 0 ) {
                     qsort( (void *) File_Info, file_count, sizeof(
                     File_Info[0] ), sort_function );
                     for ( j=0; j<file_count; j++ ) {
                            fprintf( output,
                                   '\x22%s\x22,\x22%s\x22\n',
                                   File_Info[j]->name, File_Info[j]-
                                   >desc );
                     }
              }
              fclose( output );
              file_count = 0;
       }

       Window_Restore( &Main_Window, screen );
       window(1,1,80,25);
       gotoxy( TEXT.curx, TEXT.cury );
       return(0);
}


File_Stats *
strucalloc() {
       File_Stats *temp;

       if (( temp = (File_Stats *) malloc( sizeof(File_Stats) ) ) ==
              NULL )
       die('Ran out of memory during \'malloc\' call.\n');
       return ( temp );
}

#pragma warn -sus
int sort_function( const void *a, const void *b ) {
       unsigned long code_1=0, code_2=0;
       File_Stats **case1;
       File_Stats **case2;

       case1 = (File_Stats*)a;
       case2 = (File_Stats*)b;
       code_1 = Get_Sort_Code( (*case1)->name );
       code_2 = Get_Sort_Code( (*case2)->name );
       if (code_1 < code_2 ) return( -1 );
       if (code_1 == code_2 ) return( 0 );
       return( 1 );
}
#pragma warn +sus

unsigned long
Get_Sort_Code( char *filename ) {
       unsigned long code;
       if (filename[0] == 'U' || filename[0] == 'u' ) {
                     code = 200000L;
                     code += 1000L  * (unsigned long ) (
                            filename[3] - '0' );
                     code += 10L * (unsigned long)( (filename[5]-
                            '0')*10+(filename[6]-'0') );
                     if (toupper(filename[8]) == 'D' )
                            ++code;
              }
              else { /* Recovery file name */
                     code = 100000L;
                     code += 10L  * (unsigned long ) ( (filename[3] -
                            '0')*10 + (filename[4] - '0') );
                     code += 1000L * (unsigned long)( (filename[6]-
                            '0')*10+(filename[7]-'0') );
                     if (toupper(filename[8]) == 'D' )
                            ++code;
              }
       return ( code );
}



char *
Find_Desc( char *filename ) {
       int  j, len,i=0;
       char *pointer;
```

```
        static char *empty="";

        len = strlen( filename );
        for ( j=0; j <= fcount; j++ ) {
                if (strncmpi(filename, file_desc[j], len ) == 0 ) {
                        pointer = file_desc[j] + len;
                        while (pointer[i] == ' ')
                                i++;
                        return ( pointer + i );
                }
        }
        return ( empty );
}


void
Highlight( struct Window_Style Window ) {
        textcolor( Window.backcolor );
        textbackground( Window.textcolor );
}


void
Normal( struct Window_Style Window ) {
        textcolor( Window.textcolor );
        textbackground( Window.backcolor );
}

void
activate_window( struct Window_Style Window ) {
        window( 1,1,80,25);
        window( Window.x1+1, Window.y1+1, Window.x2-1,
                Window.y2-1 );
        textcolor( Window.textcolor );
        textbackground( Window.backcolor );
        return;
}

char *
getFileName( char *Extension ) {
        char *Search;
        int done;

        Search = allocate( 129 );
        strcpy( Search, datapath );
        strcat( Search, "*." );
        strcat( Search, Extension );
        if ( start ) {
                        done = findfirst( Search, &ffblk, 0 );
                        start = 0;
                }
           else
                done = findnext( &ffblk );
        if ( done )
                return( NULL );
        free( Search );
        return (strstore( ffblk.ff_name ));
}
```

J. RIAKILL.EXE - Deletes raw RIA data files that are no longer needed.

1. *Header File for RIAKILL.C*

```
#ifndef _RIAKILL_H_
#define _RIAKILL_H_
// *-----------------------------------------*
// | SOURCE FILE: RIAKILL.H                  |
// |-----------------------------------------|
// | Header file for RIAKILL.C - File deletion |
// |                                         |
// | RIA Program - Version 1.1               |
// *-----------------------------------------*
#define RIGHT   77
#define LEFT    75
#define DELETE 83
#define HOME    71
#define END     79
#define CR      13
#define LF      10
#define SPACE   32
#define F10     68
#define UP      72
#define DOWN    80

#define LINE    0
#define ALL     1
#define NONE    2

struct File_Data {
        char *name[150];
        char *desc[150];
        char tag[150];
};

typedef struct File_Stats {
        char *name;
        char *desc;
        char tag;
} File_Stats;


struct Window_Style Main_Window = {RED,  CYAN, RED,1,
        1, 80, 25, BLACK,  CYAN, 'RIAkill (1.0) - Deletes temp LSC
        files for next assay.' };
struct Window_Style File_Window = {WHITE,  BLUE, WHITE,
        5,  2, 73, 24, WHITE,  RED  , 'Use Arrow Keys andspace
        bar to select files. (* selects all)' };

void activate_window( struct Window_Style Window );
void Delete_File( FILE *, char * );
void Describe( char *, int );
char * Find_Desc( char * );
int get_key( void );
char * getNextFile( char * );
unsigned long Get_Sort_Code( char * );
void Highlight( struct Window_Style );
void Normal( struct Window_Style );
void Print( File_Stats * );
void Print_Stats( int , char * );
void Tag( void );
int Scroll_Up (void );
int Scroll_Down (void );
int sort_function( const void *, const void * );
File_Stats * strucalloc( void );
void Update_Window( int );
#endif
```

2. *Source for RIAKILL Module*

```
// *-----------------------------------------*
// | SOURCE FILE: RIAKILL.C                  |
// |-----------------------------------------|
```

```
// | Presents the raw data RIA files to the user          |
// | and allows them to delete those that have been       |
// | used and are no longer needed for calculations.      |
// *------------------------------------------------------*
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>
#include <ctype.h>
#include <dir.h>
#include <process.h>
#include <alloc.h>

#include "rialib.h"
#include "riakill.h"


char datapath[120]="C:\\LOTUS\\LSC\\";
struct ffblk ffblk;
File_Stats *File_Info[150];
char *extensions[] = { "unk", "rec", "dte" };
int file_count=0;
int start = 1;
char Error_Msg[80];
char *file_desc[150];
int fcount=-1;
int   Window_Start,
      win_line,
      win_length,
      file_line;
extern struct text_info TEXT;
struct Window_Style *screenWindows[]={ &Main_Window,
      &File_Window, NULL };
char  Prompt_Disk=0;


int
main (int argc /*, char *argv[]*/ ) {
      char *tmp_desc, ch;
      char Filename[80];
      int  i, Update;

      gettextinfo( &TEXT );
      if ((tmp_desc = getenv("RIAPATH")) != NULL )
            strcpy( datapath, tmp_desc );
      if (argc > 1 ) {
            printf("RiaKill.Exe - Presents the RIA data files to the
                  user and displays\n");
            printf("              the descriptions, and allows the
                  user to select the.\n\n");
            printf("              files which need to be deleted.
                  \n\n");
            printf(".  It will search for files  on the default drive,
                  in the directory\n");
            printf("   \\LOTUS\\LSC, unless the environment
                  variable RIAPATH is set.\n");
            exit(0);
      }
      setcbrk( 1 );
      Read_Descriptions( datapath, &fcount, file_desc );
      File_Info[file_count] = strucalloc();
      for (i = 0; i < 3; i++ ) {
            start = 1;
            while ( ( File_Info[file_count]->name = getNextFile(
                  extensions[i] ) ) != NULL ) {
                  File_Info[file_count]->desc = Find_Desc(
                  File_Info[file_count]->name );
                  File_Info[file_count]->tag = 0;
                  ++file_count;
                  File_Info[file_count] = strucalloc();
            }
      }
      if ( file_count == 0 ) {
            textattr( TEXT.attribute );
            clrscr();
            printf( "No temporary LSC data files found.\n");
            exit(0);
      }

      qsort( (void *) File_Info, file_count, sizeof( File_Info[0] ),
            sort_function );
      if (file_count >= 21)
            {
                  File_Window.y1 = 2;
                  File_Window.y2 = 24;
            }
        else
            {
                  File_Window.y1 = ( 21 - file_count )/2 + 2;
                  File_Window.y2 =
                  File_Window.y1+file_count+1;
            }
      Screen_Setup( screenWindows );
      window(1,1,80,25);
      textcolor( Main_Window.titlefore );
      textbackground( Main_Window.titleback );
      gotoxy( 1, 25);
      Center_Text("|Push the F10 key to begin deleting selected
            files|", Main_Window );
      activate_window( File_Window );
      _wscroll = 0;
      win_length = File_Window.y2 - File_Window.y1 -1;
      for (i = 0; i< win_length; i++ ) {
            gotoxy( 1, i+1 );
            Print( File_Info[i]);
      }
      _wscroll = 1;
      gotoxy( 1, 1);
      Highlight( File_Window );
      Print( File_Info[0] );
      win_line = 1;
      file_line = 0;
      Window_Start = 0;
      gotoxy( 1, 1);
      Update_Window( ALL );
      while ( ( ch = get_key() ) != F10 ) {
            switch ( ch ) {
                  case UP:
                        Update = Scroll_Up();
                        break;
                  case SPACE:
                        Tag();
                  case DOWN:
                        Update = Scroll_Down();
                        break;
                  case '*':
                        for (i = 0; i < file_count; i++ ) {
                              File_Info[i]->tag = 1;
                        }
                        Update = ALL;
                        break;
                  default:
                        Beep();
                        Update = NONE;
                        break;
```

```
            }
            Update_Window( Update );
        }
        window(1,1,80,25);
        _wscroll=1;
        for (i = 0; i < file_count; i++ ) {
            if ( File_Info[i]->tag ) {
                strcpy(Filename, datapath );
                strcat(Filename, File_Info[i]->name);
                unlink( Filename );
            }
        }
        clrscr();
        return(0);
} // END MAIN

int
get_key( void ) {
        int ch;
        if ((ch = getch()) == 0 )
            {
                ch = getch();
                switch (ch) {
                    case F10:
                    case UP:
                    case DOWN:
                        return( ch );
                    default:
                        return( 0 );
                }
            }
        else
            {
                switch(ch) {
                    case SPACE:
                    case '*':
                        return( ch );
                    default:
                        return(0);
                }
            }
} // End get_key


int
Scroll_Up( void ) {
        if (win_line == 1 )
            if ( file_line == 0 )
                {
                    file_line = file_count - 1;
                    win_line = win_length;
                    Window_Start = file_count > 21 ?
                        file_count - 21 : 0;
                    return( ALL );
                }
              else
                {
                    --file_line;
                    --Window_Start;
                    return( ALL );
                }
          else
            {
                Normal( File_Window );
                Update_Window( LINE );
                --win_line;
                --file_line;
                Highlight( File_Window );
                Update_Window( LINE );
                return ( NONE );
            }
} // End Scroll_Up


int
Scroll_Down( void ) {
        if (win_line == win_length )
            if ( file_line == file_count - 1 )
                {
                    file_line = 0;
                    win_line = 1;
                    Window_Start = 0;
                    return( ALL );
                }
              else
                {
                    ++file_line;
                    ++Window_Start;
                    return( ALL );
                }
          else
            {
                Normal( File_Window );
                Update_Window( LINE );
                ++win_line;
                ++file_line;
                Highlight( File_Window );
                Update_Window( LINE );
                return ( NONE );
            }
} // End Scroll_Down


void
Update_Window( int code ) {
        int i;

        char morech[2];
        if ( code == NONE ) {
            gotoxy(1, win_line );
            return;
        }
        if ( code == LINE ) {
            gotoxy( 1, win_line );
            _wscroll = 0;
            Print( File_Info[file_line] );
            _wscroll = 1;
        }
        if (code == ALL ) {
            if (Window_Start != 0 )
                {
                    textattr( RED*16+LIGHTGRAY);
                    strcpy(morech,"*");
                }
              else
                {
                    textcolor( File_Window.forecolor );
                    textbackground(File_Window.backcolor);
                    strcpy(morech,"-");
                }
            window (1,1,80,25);
            gotoxy( File_Window.x1+1, File_Window.y1 );
            cputs( morech );
            activate_window( File_Window );
            if (Window_Start+win_length < file_count )
                {
```

```
                textattr( RED*16+LIGHTGRAY );
                strcpy(morech,"*");
            }
          else
            {
                textcolor( File_Window.forecolor );
                textbackground(File_Window.backcolor);
                strcpy(morech,"-");
            }
        window(1,1,80,25);
        gotoxy( File_Window.x1+1, File_Window.y2 );
        cputs( morech );
        activate_window( File_Window );
        Normal( File_Window );
        _wscroll=0;
        for ( i = 0; i<win_length; i++ ) {
            gotoxy(1, i+1 );
            Print( File_Info[Window_Start + i] );
        }
        Highlight( File_Window );
        gotoxy( 1, win_line );
        Print( File_Info[file_line]);
        _wscroll=1;
    }
} // End Update_Window

void
Tag( void ) {
    File_Info[file_line]->tag ^= 1;
}

void
Print( File_Stats *File ) {
    cprintf("  %c %-12s  %c%-40s          ",File->tag==0 ? ' ':'*',
        File->name, '"' , File->desc==NULL ? ' ':File->desc );
}

File_Stats *
strucalloc() {
    File_Stats *temp;

    if (( temp = (File_Stats *) malloc( sizeof(File_Stats) ) ) ==
        NULL )
        die("Ran out of memory during \'malloc\' call.\n");
    return ( temp );
}

#pragma warn -sus
int sort_function( const void *a, const void *b ) {

    unsigned long code_1=0, code_2=0;
    File_Stats **case1;
    File_Stats **case2;

    case1 = (File_Stats*)a;
    case2 = (File_Stats*)b;
    code_1 = Get_Sort_Code( (*case1)->name );
    code_2 = Get_Sort_Code( (*case2)->name );
    if (code_1 < code_2 ) return( -1 );
    if (code_1 == code_2 ) return( 0 );
    return( 1 );
}
#pragma warn +sus

unsigned long
Get_Sort_Code( char *filename ) {
  unsigned long code;
  if (filename[0] == 'U' || filename[0] == 'u' ) {
        code = 200000L;
        code += 1000L  * (unsigned long ) ( filename[3] - '0' );
        code += 10L * (unsigned long)( (filename[5]-
        '0')*10+(filename[6]-'0') );
        if (toupper(filename[8]) == 'D' )
          ++code;
      }
    else { /* Recovery file name */
        code = 100000L;
        code += 10L  * (unsigned long ) ( (filename[3] - '0')*10 +
          (filename[4] - '0') );
        code += 1000L * (unsigned long)( (filename[6]-
          '0')*10+(filename[7]-'0') );
        if (toupper(filename[8]) == 'D' )
          ++code;
      }
  return ( code );
}


char *
Find_Desc( char *filename ) {
  int  j, len,i=0;
  char *pointer;

  len = strlen( filename );
  for ( j=0; j <= fcount; j++ ) {
    if (strncmpi(filename, file_desc[j], len ) == 0 ) {
        pointer = file_desc[j] + len;
        while (pointer[i] == ' ') i++;
        return ( pointer + i );
    }
  }
  return ( NULL );
}


char *
getNextFile( char *Extension ) {
  char *Search;
  int done;

  Search = allocate( 129 );
  strcpy( Search, datapath );
  strcat( Search, "*." );
  strcat( Search, Extension );
  if ( start ) {
      done = findfirst( Search, &ffblk, 0 );
      start = 0;
    }
   else
    done = findnext( &ffblk );
  if ( done )
    return( NULL );
  free( Search );
  return (strstore( ffblk.ff_name ));
}


void
Highlight( struct Window_Style Window ) {
  textcolor( Window.backcolor );
  textbackground( Window.textcolor );
}
```

```
void
Normal( struct Window_Style Window ) {
   textcolor( Window.textcolor );
   textbackground( Window.backcolor );
}

void
activate_window( struct Window_Style Window ) {
   window( 1,1,80,25);
   window( Window.x1+1, Window.y1+1, Window.x2-1,
      Window.y2-1 );
   textcolor( Window.textcolor );
   textbackground( Window.backcolor );
   return;
}
```

K. Raw Data Editor - FIXIT.EXE

1. *Main body of FIXIT.BAS*

```
'*-------------------------------------------*
'| SOURCE FILE: FIXIT.BAS                    |
'|-------------------------------------------|
'| Program for editing Raw data files from the |
'| LSC. Compiles using TurboBasic 1.0 and    |
'| PowerBasic 2.0-3.1                        |
'*-------------------------------------------*
DEFINT A-Z
DIM ColorPalette(10,2)
DIM FileData(3000), FileName$(40), FileBreak%(20)
$INCLUDE 'SCREEN.INC'

'*-------------------------------------------*
'| PROCEDURE: Main                           |
'|-------------------------------------------|
'| PURPOSE: Loop and process files until done. |
'*-------------------------------------------*
Finished%=0
CALL DefineColors
CALL DefineCurrentPalette
WHILE NOT FINISHED
      CALL ChooseFileType(Extension$)
      IF Extension$='REC' THEN
            Head$='Edit Recovery Files'
      ELSE
            Head$='Edit Unknown Files'
      END IF
      CALL ScreenSetUp(Head$,7,3,2,32)
      CALL WaitDraw(10,30,10,10)
      A$ = 'C:\LOTUS\LSC\PROGRAMS\RIADIR.EXE '
      SHELL A$
      CALL GetFileNames(Extension$)
      CALL ChooseAFile
      CALL GetData(Extension$)
      CALL EditFile
      CALL StoreTheData
      CALL ScreenSetUp('LSC File Editor',7,3,2,32)
      LineNumber% = 0 : ColumnNumber%=0
      CALL DrawWindow(3,43,4)
      CALL CenterText(12,'Do you want to edit another file?
            (Y/N)',2)
      CALL CheckKey(PushedKey$)
      IF PushedKey$<>'y' AND PushedKey$<>'Y' THEN
            Finished%=-1
      END IF
WEND
CLS
END


SUB DefineCurrentPalette
      SHARED Black, Blue, Green, Cyan, Red, Magenta, Brown,
            White, Gray
      SHARED LightBlue, LightGreen, LightCyan, LightRed,
            LightMagenta
      SHARED Yellow, HighIntensityWhite, ColorPalette%()
      CALL DefinePalette(1,HighIntensityWhite,Cyan)
      CALL DefinePalette(2,White,Blue)
      CALL DefinePalette(3,Red,Cyan)
      CALL DefinePalette(4,Yellow,Blue)
      CALL DefinePalette(5,Brown,Cyan)
      CALL DefinePalette(6,Magenta,Cyan)
      CALL DefinePalette(7,LightRed,Cyan)
      CALL DefinePalette(8,Blue,Yellow)
      CALL DefinePalette(9,Yellow,Cyan)
      CALL DefinePalette(10,Red+16,White)
END SUB

'*-------------------------------------------*
'|          Subroutine - ChooseFileType      |
'|-------------------------------------------|
'| This subroutine allows the user to choose whether a |
'| Recovery or an Unknown file will be edited. |
'| Extension$ is set to REC if the file is a recovery file |
'| and if UNK the file is an unknown file.   |
'*-------------------------------------------*
SUB ChooseFileType(Extension$)
      SHARED LineNumber%, ColumnNumber%
      CALL ScreenSetUp('LSC File Editor',7,3,2,32)
      LineNumber%=8 : ColumnNumber%=13
      CALL DrawWindow(5,40,4)
      LOCATE 9,15
      PRINT 'Use the Left and Right Arrow Keys to';
      LOCATE 10,15 : PRINT 'Highlight the type of file to edit.';
      LOCATE 11,15 : PRINT 'Then push RETURN.'
      LineNumber%=15 : ColumnNumber%=13
      CALL DrawWindow(3,40,4)
      Code%=1
      CALL ExtensionDisplay(Code%)
      ChooseExtension:
      CALL CheckKey(KeyPushed$)
      IF LEN(KeyPushed$)>1 THEN GOTO TestExtendedASCII
      IF KeyPushed$=CHR$(13) THEN GOTO ExtensionChosen
      NotValidKey:
      BEEP : GOTO ChooseExtension
      TestExtendedASCII:
      ExtendCode%=ASC(MID$(KeyPushed$,2,1))
      IF ExtendCode%<>75 AND ExtendCode%<>77 THEN
            GOTO NotValidKey
      IF Code%=2 THEN Code%=1  ELSE Code%=2
      CALL ExtensionDisplay(Code%)
      GOTO ChooseExtension
      ExtensionChosen:
      IF Code%=1 THEN Extension$='REC' ELSE
            Extension$='UNK'
END SUB


'*-------------------------------------------*
'|          Subroutine - ExtensionDisplay    |
'|-------------------------------------------|
'| This subroutine diplays the words Recovery and |
'| Unknown. If Code%=1 then Recovery is in reverse |
'| video and if Code%=2 then unknown is in reverse |
'| video.                                    |
```

```
'*-------------------------------------------------------*
SUB ExtensionDisplay(Code%)
    IF Code%=2 THEN GOTO UnknownHighlighted
    RecoveryColor%=8
    UnknownColor%=2
    GOTO Display
    UnknownHighlighted:
    RecoveryColor%=2
    UnknownColor%=8
    Display:
    LOCATE 16,21
    CALL SetColor (RecoveryColor%) : PRINT 'Recovery';
    LOCATE 16,38
    CALL SetColor (UnknownColor%) : PRINT 'Unknown ';
END SUB

'*-------------------------------------------------------*
'|            Subroutine - GetFileNames                  |
'|-------------------------------------------------------|
'| This subroutine displays the file names on the screen |
'| and allows the user to select one of them.            |
'*-------------------------------------------------------*
SUB GetFileNames(Extension$)
    SHARED FileCount%,FileName$()
    IF Extension$ = 'REC' THEN
        FileName$ = 'C:\LOTUS\LSC\RFILES.DAT'
    ELSE
        FileName$ = 'C:\LOTUS\LSC\UFILES.DAT'
    END IF
    COLOR 15,0
    OPEN FileName$ FOR INPUT AS #1
    FileCount% = 0 : NumberOfFiles% = 0 : Flag=0
    WHILE NOT EOF(1)
    FileCount% = FileCount% + 1
    IF FileCount%>39 THEN
        PRINT 'Too many files in data directory (>40)'
        SYSTEM
    END IF
    LINE INPUT #1, A$
    x = 1
    B$ = MID$(A$,x,1)
    start%=0
    WHILE x<=LEN( A$ )
        IF B$=CHR$(34) THEN
            IF start% THEN
                EXIT LOOP
                ELSE
                start%=x+1
            END IF
        END IF
        x = x + 1
        IF x > LEN(A$) THEN
            EXIT LOOP
        END IF
        B$ = MID$(A$,x,1)
    WEND
    FileName$(FileCount%) = MID$( A$, start%, x-start% )
    WEND
    NumberOfFiles% = FileCount%
    GOTO SKipSOrt
    IF Extension$='REC' THEN EndPoint%=8 ELSE
        EndPoint%=7
    FOR Record% = 1 TO FileCount%
        IF LEFT$(FileName$(Record%),3)<>Extension$
        THEN GOTO EndOfRecordLoop
        NumberOfFiles% = NumberOfFiles% + 1
        FileName$(NumberOfFiles%) =
        LEFT$(FileName$(Record%),EndPoint%)+'.'+Extensi
        on$
        EndOfRecordLoop:
    NEXT Record%
    Sort the file Names [Two Pass]
    StartSort:
    BeginPoint1%=7 : NumberCharacters1%=2
    BeginPoint2%=4 : NumberCharacters2%=2
    IF Extension$='REC' THEN GOTO RECStart
    BeginPoint1%=4 : NumberCharacters1%=1
    BeginPoint2%=6 : NumberCharacters2%=2
    RECStart:
    FOR StartValue% = 1 TO NumberOfFiles%-2
        FOR FileNumber% = StartValue% TO
            NumberOfFiles%-1
            Code1% = VAL(MID$(
                FileName$(FileNumber%),
                BeginPoint1%, NumberCharacters1%))
            Code2% = VAL(
                MID$(FileName$(FileNumber%),
                BeginPoint2%, NumberCharacters2%))
            Code3% = VAL(
                MID$(FileName$(FileNumber%+1),
                BeginPoint1%, NumberCharacters1%))
            Code4% = VAL(
                MID$(FileName$(FileNumber%+1),
                BeginPoint2%, NumberCharacters2%))
            IF Code3%>Code1% THEN GOTO
                EndOfFileNumberLoop
            IF Code3%<Code1% THEN GOTO SwapValues
            IF Code4%>Code2% THEN GOTO
                EndOfFileNumberLoop
        SwapValues:
            SWAP FileName$(FileNumber%),
            FileName$(FileNumber%+1)
        EndOfFileNumberLoop:
        NEXT FileNumber%
    NEXT StartValue%
    IF Flag=1 THEN GOTO EndOfThisSubroutine
    Flag=1
    GOTO StartSort
    EndOfThisSubroutine:
    SkipSort:
    FileCount% = NumberOfFiles%
END SUB


'*-------------------------------------------------------*
'|        Subroutine - DeleteDataFromFile                |
'| This Subroutine deletes a specified number of records |
'| from the data matrix.                                 |
'*-------------------------------------------------------*
SUB DeleteDataFromFile(NumberToDelete%, RecordNumber%)
    SHARED FileData(), NumberOfRecords%
    IF NumberOfRecords%-
        NumberToDelete%<RecordNumber% THEN GOTO
        EndDeleteFromFile
    FOR Record% = RecordNumber% TO NumberOfRecords%-
        NumberToDelete%
    FileData(Record%) =
        FileData(Record%+NumberToDelete%)
    NEXT Record%
    NumberOfRecords% = NumberOfRecords% -
        NumberToDelete%
    IF NumberOfRecords%<0 THEN NumberOfRecords%=0
    EndDeleteFromFile:
END SUB
```

```
'*-------------------------------------------------------*
'|          Subroutine - AddDataToFile                   |
'|  This Subroutine adds one or two records to the data  |
'|  matrix.                                              |
'*-------------------------------------------------------*
SUB AddDataToFile(NumberToAdd%, RecordNumber%,A,B)
    SHARED FileData(), NumberOfRecords%
    FOR Record% = NumberOfRecords% TO RecordNumber%
        STEP -1
    FileData(Record%+NumberToAdd%) = FileData(Record%)
    NEXT Record%
    NumberOfRecords% = NumberOfRecords% +
        NumberToAdd%
    FileData(RecordNumber%) = A
    IF NumberToAdd%<>2 THEN GOTO EndAddDataToFile
    FileData(RecordNumber%+1) = B
    EndAddDataToFile:
END SUB

SUB AddFileBreak(BreakInFile%,Increment%)
    SHARED FileBreak%(), NumberFileBreaks%
    NumberFileBreaks% = NumberFileBreaks% + 1
    FileBreak%(NumberFileBreaks%) = BreakInFile%
    CALL SortFileBreaks
    CALL FindBreak(BreakInFile%,BreakNumber%)
    CALL MoveFileBreaks(BreakNumber%+1,Increment%)
END SUB

SUB MoveFileBreaks(Begin%,Increment%)
    SHARED NumberFileBreaks%,FileBreak%()
    IF Begin%>NumberFileBreaks% THEN GOTO
        EndMoveFileBreaks
    FOR Break% = Begin% TO NumberFileBreaks%
    FileBreak%(Break%) = FileBreak%(Break%) + Increment%
    NEXT Break%
    EndMoveFileBreaks:
END SUB

SUB DeleteFileBreak(BreakInFile%,Increment%)
    SHARED NumberFileBreaks%, Filebreak%()
    CALL FindBreak(BreakInFile%,BreakNumber%)
    FOR Break% = BreakNumber% TO NumberFileBreaks%
    FileBreak%(Break%) = FileBreak%(Break%+1)
    NEXT Break%
    CALL MoveFileBreaks(BreakNumber%,Increment%)
    NumberFileBreaks% = NumberFileBreaks% - 1
END SUB

SUB FindBreak(BreakInFile%,BreakNumber%)
    SHARED NumberFileBreaks%, FileBreak%()
    FOR Break% = 1 TO NumberFileBreaks%
    IF FileBreak%(Break%)=BreakInFile% THEN GOTO
        EndFindBreak
    NEXT Break%
    PRINT 'Error - Invalid Break Line'
    STOP
    EndFindBreak:
    BreakNumber% = Break%
END SUB



SUB ChooseAFile
    SHARED FileCount%,
        FileName$(),LineNumber%,ColumnNumber%
    SHARED FileNumber%, FileToOpen$, FileOpen$
    LineNumber% = 8 : ColumnNumber% = 32
    CALL ScreenSetUp(Head$,8,3,2,32)
    CALL DrawWindow(FileCount%+2,16,4)
    LineNumber%=4 : ColumnNumber%=0
    CALL DrawWindow(3,50,4)
    CALL CenterText(5,'Highlight the desired file, then push
        RETURN',2)
    FOR FileNumber% = 1 TO FileCount%
    LOCATE FileNumber%+8,34 : PRINT
        FileName$(FileNumber%)
    NEXT FileNumber%
    FileNumber%=1
    CALL HighlightFile(1)
    CheckForKey:
    CALL CheckKey(KeyPushed$)
    IF LEN(KeyPushed$)=2 THEN GOTO ExtendedASCII
    IF KeyPushed$=CHR$(13) THEN GOTO
        ReturnKeyPushed
    KeyError:
    BEEP : GOTO CheckForKey
    ExtendedASCII:
        Code%=ASC(RIGHT$(KeyPushed$,1))
        IF Code%<>72 AND Code%<>80 THEN GOTO
            KeyError
        OldFileNumber%=FileNumber%
        IF Code%=72 THEN FileNumber%=FileNumber%-1
        IF Code%=80 THEN FileNumber%=FileNumber%+1
        IF FileNumber%>FileCount% THEN
            FileNumber%=1
        IF FileNumber%<1 THEN
            FileNumber%=FileCount%
        CALL HighlightFile(OldFileNumber%)
    GOTO CheckForKey
    ReturnKeyPushed:
    FileOpen$ = FileName$(FileNumber%)
    FileToOpen$='C:\LOTUS\LSC\'+FileName$(FileNumber%
        )
    CLOSE
    OPEN FileToOpen$ FOR INPUT AS #1
END SUB

SUB HighlightFile(OldFileNumber%)
    SHARED FileName$(),FileNumber%
    LOCATE OldFileNumber%+8,34,0 :
    CALL SetColor (2) : PRINT FileName$(OldFileNumber%);
    LOCATE FileNumber%+8,34,0
    CALL SetColor (8)
    PRINT FileName$(FileNumber%);
    CALL SetColor (2)
END SUB

SUB GetData(Extension$)
    SHARED FileData(),NumberOfRecords%, FileBreak%(),
        NumberFileBreaks%
    NumberOfRecords%=0 : NumberFileBreaks% = 0
    WHILE NOT EOF(1)
        NumberOfRecords% = NumberOfRecords% + 1
        INPUT #1, FileData(NumberOfRecords%)
        IF FileData(NumberOfRecords%)>=0 AND
            Extension$='UNK' OR NumberOfRecords%=1
            THEN GOTO EndWhileLoop
        IF FileData(NumberOfRecords%)<1000 AND
            Extension$='REC' THEN GOTO
            EndWhileLoop
        NumberFileBreaks% = NumberFileBreaks% + 1
        FileData(NumberOfRecords%+1) =
            FileData(NumberOfRecords%)
        FileData(NumberOfRecords%) = -1
```

```
            NumberOfRecords% = NumberOfRecords% + 1
            FileBreak%(NumberFileBreaks%) =
                NumberOfRecords%-1
            EndWhileLoop:
        WEND
END SUB

SUB DisplayData
        SHARED FileData(), LineNumber%, ColumnNumber%,
            NumberOfRecords%
        SHARED Start%, NumberFileBreaks%,
            FileBreak%(),Extension$
        LineNumber%=4 : ColumnNumber%=4 : Increment%=14
        IF Start%+Increment% > NumberOfRecords% THEN
            Increment%=NumberOfRecords%-Start%
        CALL DrawWindow(17,29,4)
        FileRecord% = 0
        FOR Record% = Start% TO Start%+Increment%
            CALL DisplayARecord(Record%,1)
        NEXT Record%
END SUB


SUB HighlightData
        SHARED FileData(), DataRecord%, Start%,
            OldRecordNumber%
        Flag%=1
        Beginning:
        IF Flag%=1 THEN Record%=OldRecordNumber%
        IF Flag%<>1 THEN Record%=DataRecord%
        CALL DisplayARecord(Record%,Flag%)
        IF Flag%=1 THEN Flag%=2 : GOTO Beginning
END SUB


SUB DisplayARecord(Record%,Flag%)
        SHARED FileData(), NumberFileBreaks%, Start%,
            FileBreaks%(), Extension$
        IF Flag%=1 THEN CALL SetColor(2) ELSE Call
            SetColor(8)
        LOCATE Record%-Start%+5,9,0
        IF FileData(Record%)>=0 THEN GOTO NotAFileBreak
        PRINT 'Begin New File'
        GOTO EndDisplayARecord
        NotAFileBreak:
        CALL DetermineRecordNumber(Record%,FileRecord%)
        PRINT USING '###';FileRecord%;
        PRINT ' ';
        PRINT USING '####,.######';FileData(Record%);
        EndDisplayARecord:
        CALL SetColor(2)
END SUB

SUB DetermineRecordNumber(Record%,FileRecord%)
        SHARED NumberFileBreaks%,FileBreak%(),Extension$
        IF NumberFileBreaks%=0 THEN StartRec%=0 : GOTO
            SetFileRecord
        IF Record%=1 AND Extension$<>'REC' THEN StartRec%=0
            : GOTO SetFileRecord
        FOR Break% = 1 TO NumberFileBreaks%
            IF Record%<FileBreak%(Break%) AND Break%>1
                THEN StartRec%=FileBreak%(Break%-1) :
                GOTO SetFileRecord
            IF Record%<FileBreak%(Break%) AND Break%=1
                THEN StartRec%=0 : GOTO SetFileRecord
        NEXT Break%
        StartRec% = FileBreak%(NumberFileBreaks%)
        SetFileRecord:
        FileRecord% = Record% - StartRec%
END SUB

SUB SortFileBreaks
        SHARED NumberFileBreaks%,FileBreak%()
        FOR StartPoint% = 1 TO NumberFileBreaks%-1
            FOR Break% = StartPoint% TO NumberFileBreaks%-1
                IF FileBreak%(Break%+1)>=FileBreak%(Break%)
                    THEN GOTO EndBreakSortingLoop
                SWAP FileBreak%(Break%+1),
                    FileBreak%(Break%)
                EndBreakSortingLoop:
            NEXT Break%
        NEXT StartPoint%
END SUB

SUB FindFileBreaks
        SHARED NumberFileBreaks%, FileBreak%(),
            NumberOfRecords%, FileData()
        NumberFileBreaks% = 0
        FOR Record% = 1 TO NumberOfRecords%
            IF FileData(Record%) >= 0 THEN GOTO
                ContinueLoop
            NumberFileBreaks% = NumberFileBreaks% + 1
            FileBreak%(NumberFileBreaks%) = Record%
            ContinueLoop:
        NEXT Record%
END SUB

SUB EditFile
DIM KeyCode%(10)
DATA 71,72,73,79,80,81,82,83
SHARED FileData(), NumberOfRecords%, LineNumber%,
ColumnNumber%
SHARED Start%, DataRecord%, OldRecordNumber%,
FileOpen$,Extension$
SHARED NumberFileBreaks%, FileBreak%()
IF Extension$='REC' THEN ExtCode%=2 ELSE ExtCode%=1
RESTORE
FOR Code% = 1 TO 8
READ KeyCode%(Code%)
NEXT Code%
Start%=1 : DataRecord%=1 : OldRecordNumber%=1
CALL ScreenSetUp('Editing '+FileOpen$,8,3,2,32)
LineNumber% = 4 : ColumnNumber% = 35
CALL DrawWindow(13,43,4)
LOCATE 5,37 : PRINT 'Ins   Insert before the current line.';
LOCATE 6,37 : PRINT 'Del   Delete the current line.';
LOCATE 7,37 : PRINT 'PgUp   Back data up one screen full.';
LOCATE 8,37 : PRINT 'PgDn   Advance data one screen full.';
LOCATE 9,37 : PRINT 'Home   Goto beginning of data.';
LOCATE 10,37 : PRINT 'End      Goto end of data.';
LOCATE 11,37 : PRINT 'Return  Add data to end of data.';
LOCATE 12,37 : PRINT CHR$(24);'      Goto the previous line.';
LOCATE 13,37 : PRINT CHR$(25);'      Goto the next line.';
LOCATE 14,37 : PRINT 'Esc Write data to Hard Disk.';
LOCATE 15,37 : PRINT 'Insert a -1 for a file break (UNK only).';
CALL DisplayData
CALL HighlightData
TestKey:
CALL CheckKey(KeyPushed$)
IF LEN(KeyPushed$)=2 THEN GOTO ExtendedCodes
Code% = ASC(KeyPushed$)
IF Code%=27 THEN GOTO StoreData
IF Code%=13 AND DataRecord%=NumberOfRecords% THEN
GOTO AddToEnd
IF Code%>47 AND Code%<58 OR Code%=45 THEN GOTO
ChangeData
```

```
BadKey:
    BEEP : GOTO TestKey
ExtendedCodes:
    Code%=ASC(RIGHT$(KeyPushed$,1))
    FOR Test% = 1 TO 8
        IF KeyCode%(Test%) = Code% THEN GOTO
GoodKey
    NEXT Test%
    GOTO BadKey
    GoodKey:
    SELECT CASE Test%
    CASE = 1   'Home Key
        Start%=1 : DataRecord%=1 : OldRecordNumber%=1
        CALL DisplayData
        CALL HighlightData
    CASE = 2   'Up Arrow Key
        OldRecordNumber%=DataRecord%
        DataRecord% = DataRecord% - 1
        IF DataRecord%<1 THEN BEEP : DataRecord%=1
        IF DataRecord%>=Start% THEN GOTO
            StillOnScreen
        Start%=DataRecord%
        CALL DisplayData
        StillOnScreen:
        CALL HighlightData
    CASE = 3   'Page Up Key
        LineNumber%=DataRecord%-Start%+1
        Start%=Start%-15
        IF Start%>=1 THEN GOTO NotPastBeginningOfFile
        Start%=1 : DataRecord%=LineNumber% :
            OldRecordNumber%=DataRecord%
        GOTO DisplayTheData
       NotPastBeginningOfFile:
        DataRecord%=DataRecord%-15
        OldRecordNumber%=DataRecord%
        DisplayTheData:
        Call DisplayData
        CALL HighlightData
    CASE = 4   'End Key
        IF NumberOfRecords%<15 THEN Start%=1 ELSE
            Start%=NumberOfRecords%-14
        DataRecord%=NumberOfRecords%
        OldRecordNumber% = DataRecord%
        CALL DisplayData
        CALL HighlightData
    CASE = 5   'Down Arrow Key
        OldRecordNumber% = DataRecord%
        DataRecord% = DataRecord% + 1
        IF DataRecord%>NumberOfRecords% THEN BEEP :
            DataRecord%=OldRecordNumber% : GOTO
            TestKey
        IF DataRecord%<=Start%+14 THEN GOTO
            StillOnScreen2
        Start%=Start%+1
        CALL DisplayData
        StillOnScreen2:
        CALL HighlightData
    CASE = 6   'Page Down Key
        LineNumber%=DataRecord%-Start%+1
        Start% = Start% + 15
        IF Start%+14<=NumberOfRecords% THEN GOTO
            NotPastEndOfFile
        Start% = NumberOfRecords% - 14
        IF Start%<1 THEN Start%=1
        DataRecord%=LineNumber%+Start%-1 :
            OldRecordNumber%=DataRecord%
        GOTO DisplayTheData
        NotPastEndOfFile:
            DataRecord% = DataRecord% + 15
            OldRecordNumber% = DataRecord%
            GOTO DisplayTheData
    CASE = 7   'Insert Key
        CALL AddDataToFile(1,DataRecord%,0,0)
       InsertMode:
        CALL DisplayData
        LOCATE DataRecord%-Start%+5,9,0
        CALL DetermineRecordNumber(
            DataRecord%,FileRecord%)
        CALL SetColor(8)
        PRINT USING '###';FileRecord%;
        PRINT STRING$(16,32);
        CALL SetColor (2)
        LineNumber%=17 : ColumnNumber%=42
        CALL DrawWindow(5,19,4)
        LOCATE 18,43 : PRINT 'Enter the value';
        LOCATE 19,43 : PRINT 'to insert at ';FileRecord%;
        LOCATE 20,43 : INPUT ";NewValue
        Increment%=2
       GotNewValue:
        FileData(DataRecord%) = NewValue
        IF NewValue<1000 OR Extension$<>'REC' THEN
            GOTO HighlightTheData
        IF OldValue>1000 AND Extension$='REC' THEN
            GOTO HighlightTheData
        CALL AddDataToFile(1,DataRecord%,-1,0)
        HighlightTheData:
        CALL FindFileBreaks
        CALL DisplayData
        CALL HighlightData
        LineNumber%=17 : ColumnNumber%=42
        CALL EraseWindow(5,19,2,32)
    CASE = 8   'Delete Key
        TypeCode%=1
        IF DataRecord%<NumberOfRecords% THEN GOTO
            NotLastRecord
        NumberOfRecords% = NumberOfRecords%-1
        DataRecord%=DataRecord% - 1 :
            OldRecordNumber% = DataRecord%
        GOTO CheckUp:
       NotLastRecord:
        BreakCode% = 1
        IF FileData(DataRecord%)>1000 AND
            Extension$='REC' THEN GOTO DeleteBreak
        IF FileData(DataRecord%)>=0 THEN
            NotDeleteFileBreak
        IF Extension$='REC' THEN GOTO BadKey
       DeleteBreak:
        IF ExtCode% = 2 THEN BreakCode% = 2
        NotDeleteFileBreak:
        CALL DeleteDataFromFile( BreakCode%,
            DataRecord%-BreakCode%+1)
        CheckUp:
        IF NumberOfRecords%=0 THEN GOTO StoreData
        CALL FindFileBreaks
        GOTO DisplayTheData
END SELECT
GOTO TestKey
AddToEnd:
Escape%=1
DataRecord% = DataRecord% + 1
IF DataRecord%-Start%>14 THEN Start%=DataRecord%-
    14
NumberOfRecords% = NumberOfRecords% + 1
GOTO InsertMode:

ChangeData:
```

```
        IF FileData(DataRecord%)<0 AND Extension$="REC"
            THEN GOTO BadKey
        CALL SetColor (2)
        LineNumber%=17 : ColumnNumber%=42
        CALL DrawWindow(5,17,4)
        CALL SetColor(2)
        CALL DetermineRecordNumber( DataRecord%,
            FileRecord%)
        OldValue = FileData(DataRecord%)
        LOCATE 18,43 : PRINT "New value for";
        LOCATE 19,43 : PRINT "recovery ";FileRecord%;"=";
        LOCATE 20,43 : PRINT KeyPushed$
        Number$=KeyPushed$
        ColumnNumber%=44
        GetKey:
            CALL CheckKey(KeyPushed$)
            IF LEN(KeyPushed$)>1 THEN GOTO KeyWrong
            IF KeyPushed$=CHR$(27) THEN GOTO
                FinishedChange
            ACode%=ASC(KeyPushed$)
            IF ACode%<47 OR ACode%>58 AND ACode%<>45
                AND ACode%<>44 THEN GOTO
                NotANumber:
            AddToString:
                Number$ = Number$ + KeyPushed$
                Locate 20,ColumnNumber%,1 : PRINT
                    KeyPushed$;
                ColumnNumber%=ColumnNumber%+1
                GOTO GetKey
            NotANumber:
                Code%=ASC(KeyPushed$)
                IF Code%=13 THEN GOTO GotData
                IF Code%= 8 THEN GOTO BackSpaceKey
                IF Code%=46 THEN GOTO AddToString
                KeyWrong:
                BEEP : GOTO GetKey
            GotData:
                'If the old value is a beginning of a recovery
                ' file, and the new value is less than 1000, then
                ' we must delete one of the file breaks.
                Increment%=1
                NewValue = Val(Number$)
                IF OldValue>1000 AND NewValue<=1000
                    THEN DeleteFlag%=1 ELSE
                    DeleteFlag%=0
                FinishedChange:
                ColumnNumber%=42
                CALL EraseWindow(5,17,2,32)
                GOTO GotNewValue
            BackSpaceKey:
                LOCATE 20,ColumnNumber%-1 : PRINT " ";
                ColumnNumber%=ColumnNumber%-1
                GOTO GetKey
StoreData:
END SUB

SUB MoveUpTheData
    SHARED NumberOfRecords%,Datarecord%,FileData()
    FOR RecordNumber% = NumberOfRecords% TO
        DataRecord% STEP -1
        FileData(RecordNumber%+1) =
        FileData(RecordNumber%)
    NEXT RecordNumber%
    FileData(DataRecord%) = 0.0
    NumberOfRecords% = NumberOfRecords% + 1
END SUB

SUB StoreTheData
    SHARED FileData(), NumberOfRecords%, LineNumber%,
        ColumnNumber%
    SHARED Start%, DataRecord%, OldRecordNumber%,
        FileOpen$, FileToOpen$
    SHARED FileCount%, FileName$(), Extension$
    IF NumberOfRecords%=0 THEN KILL FileToOpen$ :
        GOTO EndOfStoreTheDataSubroutine
    CALL ScreenSetUp("Writing Data To The Hard
        Disk",8,3,2,32)
    CLOSE
    IF Extension$="REC" THEN BeginPoint%=4 ELSE
        BeginPoint%=6
    OPEN FileToOpen$ FOR OUTPUT AS #1
    LPRINT FileToOpen$;" stored on hard disk."
    LineNumber% = 8 : ColumnNumber% = 8
    CALL DrawWindow(8,15,4)
    LOCATE 10,10 : PRINT FileOpen$;
    Begin% = 1 : Start%=1 : Count%=11
    StartLoop:
    FOR Record% = Begin% to NumberOfRecords%
    IF FileData(Record%)>=0 OR Start%=1 THEN GOTO
        WriteDataToDisk
    Code% = VAL(MID$(FileOpen$,BeginPoint%,2))
    AddOneToCode:
    Code% = Code% + 1
    ToInsert$ = STR$(Code%)
    ToInsert$ = RIGHT$(ToInsert$,2)
    IF Code%<10 THEN MID$(ToInsert$,1,1)="0"
    Mid$(FileOpen$,BeginPoint%,2)=ToInsert$
    FOR FileCode% = 1 TO FileCount%
        IF FileOpen$ = FileName$(FileCode%) THEN GOTO
        AddOneToCode
    NEXT FileCode%
    FileCount% = FileCount% + 1
    FileName$(FileCount%) = FileOpen$
    FileToOpen$ = "C:\LOTUS\LSC\"+FileOpen$
    CLOSE 1
    LOCATE Count%,10 : PRINT FileOpen$;
    Count% = Count% + 1
    OPEN FileToOpen$ FOR OUTPUT AS #1
    LPRINT FileToOpen$;" stored on hard disk."
    GOTO EndThisLoop
    WriteDataToDisk:
    PRINT #1, FileData(Record%)
    Start%=2
    EndThisLoop:
    NEXT Record%
    CLOSE
    EndOfStoreTheDataSubroutine:
END SUB
```

2. *Windowing Routines for Fixit.Bas*

```
'*-------------------------------------------------------*
'| These are the Computer RA Turbo Basic Include       |
'| Subroutines    for use on color monitor computers.  |
'|                                                     |
'|                    SCREEN.INC                       |
'|-----------------------------------------------------|
'|    Written by Jerry Jacobs  (January 10, 1989).     |
'*-------------------------------------------------------*

SUB DefinePalette(PaletteReg%, Foreground%, BackGround%)
    SHARED Black, Blue, Green, Cyan, Red, Magenta, Brown,
        White, Gray
    SHARED LightBlue, LightGreen, LightCyan, LightRed,
        LightMagenta
```

```
        SHARED Yellow, HighIntensityWhite, ColorPalette%()
        ColorPalette%(PaletteReg%,1) = Foreground%
        ColorPalette%(PaletteReg%,2) = Background%
END SUB


SUB SetColor(PaletteReg%)
        SHARED Black, Blue, Green, Cyan, Red, Magenta, Brown,
                White, Gray
        SHARED LightBlue, LightGreen, LightCyan, LightRed,
                LightMagenta
        SHARED Yellow, HighIntensityWhite, ColorPalette%()
        COLOR ColorPalette%(PaletteReg%,1),
                ColorPalette%(PaletteReg%,2)
END SUB



'*-------------------------------------------------------*
'|                Subroutine - CenterText                |
'|-------------------------------------------------------|
'| This subroutine centers text on a desired line and prints |
'| it in the desired color.  If the row is not specified,  |
'|    then it is set at line 12.                          |
'*-------------------------------------------------------*
SUB CenterText(Row%,Text$,TextColor%)
        SHARED Black, Blue, Green, Cyan, Red, Magenta, Brown,
                White, Gray
        SHARED LightBlue, LightGreen, LightCyan, LightRed,
                LightMagenta
        SHARED Yellow, HighIntensityWhite, ColorPalette%()
        IF Row%=0 THEN Row%=12
        LengthOfText%=LEN(Text$)
        Margin%=(80-LengthOfText%)/2
        LOCATE Row%,Margin%
        CALL SetColor (TextCOlor%)
        Print Text$;
END SUB

'*-------------------------------------------------------*
'|                Subroutine - CheckKey                  |
'|-------------------------------------------------------|
'| This subroutine waits for a key to be pushed, then returns |
'| it as KeyPushed$.                                      |
'*-------------------------------------------------------*
SUB CheckKey(KeyPushed$)
        KeyPushed$=""
        WHILE KeyPushed$=""
                KeyPushed$=INKEY$
        WEND
END SUB


'*-------------------------------------------------------*
'|                Subroutine - DefineColors              |
'|-------------------------------------------------------|
'| This subroutine defines the codes for each color for use |
'| in other subroutines, on computers with color monitors. |
'*-------------------------------------------------------*
SUB DefineColors
        SHARED Black, Blue, Green, Cyan, Red, Magenta, Brown,
                White, Gray
        SHARED LightBlue, LightGreen, LightCyan, LightRed,
                LightMagenta
        SHARED Yellow, HighIntensityWhite
        Black           = 0
        Blue            = 1
        Green           = 2
        Cyan            = 3
        Red             = 4
        Magenta         = 5
        Brown           = 6
        White           = 7
        Gray            = 8
        LightBlue       = 9
        LightGreen      = 10
        LightCyan       = 11
        LightRed        = 12
        LightMagenta    = 13
        Yellow.         = 14
        HighIntensityWhite = 15
END SUB


'*-------------------------------------------------------*
'|              Subroutine - DrawHeaderWindow             |
'|-------------------------------------------------------|
'| This subroutine draws a double bordered window with the |
'| header string centered in the upper line of the window. |
'| The window is drawn at LineNumber%,ColumnNumber%,       |
'| where this is the upper left corner of the window.  If Line- |
'| Number% is zero, then the window is centered vertically, |
'| and if ColumnNumber% is zero, then the window is centered |
'| horizontally.  The border color and header string colors |
'| are as choosen by the user.                            |
'*-------------------------------------------------------*
SUB DrawHeaderWindow(Length%, Wide%, Header$,
                HeaderColor%, BorderColor%)
        SHARED LineNumber%, ColumnNumber%
        SHARED Black, Blue, Green, Cyan, Red, Magenta, Brown,
                White, Gray
        SHARED LightBlue, LightGreen, LightCyan, LightRed,
                LightMagenta
        SHARED Yellow, HighIntensityWhite
        IF ColumnNumber%=0 THEN
                ColumnNumber% = (80-Wide%)/2
        END IF
        IF LineNumber%=0 THEN
                LineNumber%=(25-Length%)/2
        END IF
        HeaderLength%   = LEN(Header$)
        BorderLength1%  = (Wide%-HeaderLength%)/2-1
        BorderLength2%  = Wide%-BorderLength1%-
                HeaderLength%-2
        LOCATE LineNumber%,ColumnNumber%
        CALL SetColor (BorderColor%)
        PRINT  CHR$(201)+STRING$(BorderLength1%,205);
        CALL SetColor (HeaderColor%)
        PRINT Header$;
        CALL SetColor (BorderColor%)
        PRINT STRING$(BorderLength2%,205)+CHR$(187);
        FOR X% = 1 to Length%-2
                LOCATE LineNumber%+X%,ColumnNumber%
                PRINT CHR$(186)+STRING$(Wide%-
                        2,32)+CHR$(186);
        NEXT X%
        LOCATE LineNumber%+Length%-1,ColumnNumber%
        PRINT CHR$(200)+STRING$(Wide%-2,205)+CHR$(188);
END SUB


'*-------------------------------------------------------*
'|              Subroutine - DrawSingleWindow             |
'|-------------------------------------------------------|
'| This subroutine draws a single bordered window with the |
'| specified border color at the location LineNumber%,     |
```

```
'| ColumnNumber%, where this is the upper left corner of the |
'| window.  If LineNumber% is zero, then the window is cen- |
'| tered vertically, and if ColumnNumber% is zero, then the |
'| window is centered horizontally.                         |
'*----------------------------------------------------------*
SUB DrawSingleWindow(Length%,Wide%,BorderColor%)
     SHARED LineNumber%, ColumnNumber%
     SHARED Black, Blue, Green, Cyan, Red, Magenta, Brown,
          White, Gray
     SHARED LightBlue, LightGreen, LightCyan, LightRed,
          LightMagenta
     SHARED Yellow, HighIntensityWhite
     IF ColumnNumber%<>0 THEN GOTO CenterSkip3
     ColumnNumber% = (80-Wide%)/2
     CenterSkip3:
     IF LineNumber%<>0 THEN CenterSkip4
     LineNumber%=(25-Length%)/2
     CenterSkip4:
     LOCATE LineNumber%,ColumnNumber%
     CALL SetColor (BorderColor%)
     PRINT  CHR$(218)+STRING$(Wide%-2,196)+CHR$(191);
     FOR X% = 1 to Length%-2
          LOCATE LineNumber%+X%,ColumnNumber%
          PRINT CHR$(179)+STRING$(Wide%-
               2,32)+CHR$(179);
     NEXT X%
     LOCATE LineNumber%+Length%-1,ColumnNumber%
     PRINT CHR$(192)+STRING$(Wide%-2,196)+CHR$(217);
END SUB


'*----------------------------------------------------------*
'|                 Subroutine - DrawWindow                  |
'|----------------------------------------------------------|
'| This subroutine draws a double bordered window with the  |
'| specified border color at the location LineNumber%,      |
'| ColumnNumber%, where this is the upper left corner of the|
'| window.  If LineNumber% is zero, then the window is cen- |
'| tered vertically, and if ColumnNumber% is zero, then the |
'| window is centered horizontally.                         |
'*----------------------------------------------------------*
SUB DrawWindow(Length%,Wide%,BorderColor%)
     SHARED LineNumber%, ColumnNumber%
     SHARED Black, Blue, Green, Cyan, Red, Magenta, Brown,
          White, Gray
     SHARED LightBlue, LightGreen, LightCyan, LightRed,
          LightMagenta
     SHARED Yellow, HighIntensityWhite
     IF ColumnNumber%<>0 THEN GOTO SkipCenter
     ColumnNumber% = (80-Wide%)/2
     SkipCenter:
     IF LineNumber%<>0 THEN SkipCenter2
     LineNumber%=(25-Length%)/2
     SkipCenter2:
     LOCATE LineNumber%,ColumnNumber%
     CALL SetColor (BorderColor%)
     PRINT  CHR$(201)+STRING$(Wide%-2,205)+CHR$(187);
     FOR X% = 1 to Length%-2
          LOCATE LineNumber%+X%,ColumnNumber%
          PRINT CHR$(186)+STRING$(Wide%-
               2,32)+CHR$(186);
     NEXT X%
     LOCATE LineNumber%+Length%-1,ColumnNumber%
     PRINT CHR$(200)+STRING$(Wide%-2,205)+CHR$(188);
END SUB

'*----------------------------------------------------------*
'|                Subroutine - EraseWindow                  |
'|----------------------------------------------------------|
'| This subroutine erases an area by covering it with a     |
'| character of a specified color.  The area to be erased is|
'| is defined by Length% and Wide% and is located at Line-  |
'| Number%,ColumnNumber%.  If either of these is zero,      |
'| then the blanked area is centered in that dimension.     |
'*----------------------------------------------------------*
     SUB EraseWindow(Length%, Wide%, BackgroundColor%,
          BackgroundCharacterCode%)
     SHARED LineNumber%, ColumnNumber%
     SHARED Black, Blue, Green, Cyan, Red, Magenta, Brown,
          White, Gray
     SHARED LightBlue, LightGreen, LightCyan, LightRed,
          LightMagenta
     SHARED Yellow, HighIntensityWhite
     IF ColumnNumber%<>0 THEN
          ColumnNumber1% = ColumnNumber%
       ELSE
          ColumnNumber1% = (80-Wide%)/2
     END IF
     IF LineNumber%<>0 THEN
          LineNumber1% = LineNumber%
       ELSE
          LineNumber1%=(25-Length%)/2
     END IF
     LOCATE LineNumber1%,ColumnNumber1%
     CALL SetColor (BackgroundColor%)
     IF CSRLIN+Length%>25 THEN Length%=25-CSRLIN
     FOR X% = 1 to Length%
          LOCATE LineNumber1%+X%-1,ColumnNumber1%
          PRINT
          STRING$(Wide%,BackgroundCharacterCode%);
     NEXT X%
END SUB

'*----------------------------------------------------------*
'|                Subroutine - ScreenSetUp                  |
'|----------------------------------------------------------|
'| This subroutine draws a border around the screen, then   |
'| fills the screen with the background character, and      |
'| prints the title string centered in the top border.      |
'*----------------------------------------------------------*
SUB ScreenSetUp(Title$, TitleColor%, BorderColor%,
          BackGroundColor%, BackgroundCharacterCode%)
     SHARED Black, Blue, Green, Cyan, Red, Magenta, Brown,
          White, Gray
     SHARED LightBlue, LightGreen, LightCyan, LightRed,
          LightMagenta
     SHARED Yellow, HighIntensityWhite
     CLS
     CALL Low
     Length%=LEN(Title$)
     Centering%=(76-Length%)/2
     Remain%=76-Centering%-Length%
     LOCATE 1,2 ' 1 : COLOR White
     '    PRINT CHR$(32);
     CALL SetColor (BorderColor%)
     PRINT CHR$(219)+STRING$(Centering%,223);
     CALL SetColor (TitleColor%)  : PRINT Title$;
     CALL SetColor (BorderColor%)
     PRINT STRING$(Remain%,223)+CHR$(219);
     FOR X% = 2 to 24
          LOCATE X%,2  '1 : COLOR White
          PRINT CHR$(32);
          CALL SetColor (BorderColor%)
          PRINT CHR$(219);
          CALL SetColor (BackgroundColor%)
          PRINT STRING$(76,BackgroundCharacterCode%);
```

```
            CALL SetColor (BorderColor%)
            PRINT CHR$(219);
        NEXT X%
        LOCATE 25,2 ':   COLOR White    : PRINT CHR$(32);
        CALL SetColor (BorderColor%)
        PRINT CHR$(219)+STRING$(76,220)+CHR$(219);
END SUB


'*-------------------------------------------------*
'|              Subroutine - WaitDraw              |
'|-------------------------------------------------|
'| This subroutine draws the figure :  *-----------*|
'|                                     | Please Wait. |
'|                                     *-----------*|
'| on the screen at the location X%,Y%, where X%,Y% is the |
'| upper left corner of the box. The words 'Please Wait.' |
'| blink.                                          |
'*-------------------------------------------------*
SUB WaitDraw(X%,Y%,TextColor,BorderColor)
        SHARED Black, Blue, Green, Cyan, Red, Magenta, Brown,
            White, Gray
        SHARED LightBlue, LightGreen, LightCyan, LightRed,
            LightMagenta
        SHARED Yellow, HighIntensityWhite
        CALL SetColor (BorderColor)
        IF X%=0 THEN X%=11
        IF Y%=0 THEN Y%=32
        LOCATE X%, Y%
        PRINT CHR$(218)+STRING$(15,196)+CHR$(191)
        LOCATE X%+1,Y%
        PRINT CHR$(179)+STRING$(15,32)+ CHR$(179)
        LOCATE X%+2,Y%
        PRINT CHR$(192)+STRING$(15,196)+CHR$(217)
        LOCATE X%+1,Y%+2 :CALL SetColor (TextColor)
        PRINT 'Please Wait.';
END SUB


'*-------------------------------------------------*
'|    Subroutine - WaitErase                       |
'*-------------------------------------------------*
'| This subroutine erases the figure draw by the WaitDraw |
'| subroutine, covering it over with the character(ASCII 176). |
'*-------------------------------------------------*
SUB WaitErase(X%, Y%, BackgroundColor,
            BackgroundCharacterCode%)
        SHARED Black, Blue, Green, Cyan, Red, Magenta, Brown,
            White, Gray
        SHARED LightBlue, LightGreen, LightCyan, LightRed,
            LightMagenta
        SHARED Yellow, HighIntensityWhite
        CALL SetColor (BackgroundColor)
        LOCATE X%,Y%
        PRINT STRING$(20,BackgroundCharacterCode%);
        LOCATE X%+1,Y%
        PRINT STRING$(20,BackgroundCharacterCode%);
        LOCATE X%+2,Y%
        PRINT STRING$(20,BackgroundCharacterCode%);
END SUB

'*-------------------------------------------------*
'|    Subroutines :  Hi                            |
'|                   Low                           |
'|                   Reverse                       |
'|                   ReverseLow                    |
'|                   ReverseBlink                  |
'|                   Blink                         |
'|-------------------------------------------------|
'| These short subroutines are for use on monochrome screen |
'| computers, but can be used with computers equipped with |
'| color monitors. They control the way the characters are |
'| printed on the screen.                          |
'*-------------------------------------------------*
SUB Hi
        COLOR 14,1
END SUB

SUB Low
        COLOR 15,1
END SUB

SUB Reverse
        COLOR 1,7
END SUB

SUB ReverseLow
        COLOR 1,8
END SUB

SUB ReverseBlink
        COLOR 17,7
END SUB

SUB Blink
        COLOR 31,1
END SUB
```

APPENDIX C. - DIVERSE: A Utility for calculating Finite State Diversities

1. General Description:

The DIVERSE utility is a Microsoft EXCEL worksheet which uses macros to calculate the diversity indices of Shannon (Shannon and Weaver, 1949) and Brillouin (Brillouin, 1962) for data sets. For a detailed discussion of the indices, see Pielou, 1969.

2. Macro Listing (DIVERSE.XLM)

	A	B
	DiversityCalc	=RESULT(1)
		=ARGUMENT('dataField',8)
		=VOLATILE(TRUE)
	SummedArea	=SUM(dataField)
		=IF(NOT(SummedArea))
		= RETURN(0)
		=END.IF()
		=SET.VALUE(SummedDiversity,0)
		=FOR.CELL('CurrentCell',dataField,TRUE)
		=IF(NOT(ISTEXT(CurrentCell)))
	Proportion	= CurrentCell/SummedArea
		= IF(Proportion<>0)
		= SET.VALUE(SummedDiversity, SummedDiversity-(Proportion)*LN(Proportion))
		= END.IF()
		=END.IF()
		=NEXT()
		=RETURN(DIVERSE.XLM!SummedDiversity)

RecalculateTable	=CALCULATE.NOW()
	=CALCULATE.NOW()
	=RETURN()
EmptyTable	=SELECT(!$B$5:$M$16)
	=CLEAR(3)
	=SELECT(!$B$5)
	=RETURN()
SummedDiversity	
SummedCount	
ProductedDiversity	
H	
BrillouinDiversityCalc	=RESULT(1)
	=ARGUMENT('dataField',8)
	=VOLATILE(TRUE)
SummedBArea	=SUM(dataField)
Factorial	=FACT(SummedBArea)
	=SET.VALUE(ProductedDiversity,1)
	=FOR.CELL('CurrentCell',dataField,TRUE)
	= SET.VALUE(ProductedDiversity, ProductedDiversity*FACT(CurrentCell))
	=NEXT()
	=SET.VALUE(H,1/SummedBArea*LN(Factorial/ProductedDiversity))
	=RETURN(DIVERSE.XLM!H)

		=FACT(3)
	Auto_Open	=VOLATILE(TRUE)
	Path	=GET.DOCUMENT(2)
		=HIDE()
		=OPEN(Path&"\DIVERSE.XLS")
		=WINDOW.MAXIMIZE("DIVERSE.XLS")
		=SELECT(!$B5)
		=RETURN()

3. Worksheet Design (DIVERSE.XLS)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
2																			
3																			
4			Interval																
5		State	1	2	3	4	5	6	7	8	9	10	11	12	Weighted Diversity of Invervals (Shannon)	Weighted Diversity of Invervals (Brillouin)			
6		1															Total Diversity	0	
7		2															Index	1.000	
8		3															Hmax		
9		4															H'		
10		5															X	0	
11		6															Y	1	
12		7															s	12	
13		8															r	0	
14		9															H		
15		10																	
16		11																Shannon	Brillouin
17		12																	
18		Clear Table															Mean State Interval Diversity	0.000	0.000
19		Proportion														Calculate Table	Equally Good Intervals		
20		Interval Totals	0	0	0	0	0	0	0	0	0	0	0	0			Temporal State Diversity		
21																	Temporal Evenness	0.000	

The "Clear Table" button is linked to the EmptyTable Function, and the Calculate Table" button is linked to the RecalculateTable Function.

4. Macro Sheet Named Cells (DIVERSE.XLM)

Name	Reference	Type
Auto_Open	=$B$54	Function
BrillouinDiversityCalc	=$B$38	Function
CurrentCell	='DIVERSE.XLS'!$O$19	Reference
dataField	='DIVERSE.XLS'!$B$18:$M$18	Argument
Diversity	=$B$10	Variable
DiversityCalc	=$B$2	Funtion
EmptyTable	=$B$25	Function
Factorial	=$B$42	Variable
H	=$B$35	Variable
Path	=$B$55	Variable
ProductedDiversity	=$B$34	Variable
Proportion	=$B$12	Variable
RecalculateTable	=$B$21	Function
SummedArea	=$B$5	Variable
SummedBArea	=$B$41	Variable
SummedCount	=$B$33	Variable
SummedDiversity	=$B$32	Variable

4. Worksheet Named Cells (DIVERSE.XLS)

Name	Reference	Type
dataArea	=$B$5:$M$16	Reference
Diversity	=$Q$6	Variable
EGI	=$Q$18	Variable
EGI_B	=$R$18	Variable
H	=$Q$14	Variable
Hmax	=$Q$8	Variable
Hprime	=$Q$9	Variable
Intervals	=$B$4:$M$4	Variable
MSTD	=$Q$17	Variable
MSTD_B	=$R$17	Variable
r_	=$Q$13	Variable
s	=$Q$12	Variable
Total	=$Q$5	Variable
X	=$Q$10	Variable
Y	=$Q$11	Variable

5. Worksheet Cell Functions

	N	O
	Weighted Diversity of Invervals (Shannon)	Weighted Diversity of Invervals (Brillouin)
	=IF(Total,SUM(B5:M5)/Total*EXP('DIVERSE.XLM'!DiversityCalc(B5:M5)),"")	=IF(SUM(B5:M5),SUM(B5:M5)/Total*EXP('DIVERSE.XLM'!BrillouinDiversityCalc(B5:M5)),"")
	=IF(Total,SUM(B6:M6)/Total*EXP('DIVERSE.XLM'!DiversityCalc(B6:M6)),"")	=IF(SUM(B6:M6),SUM(B6:M6)/Total*EXP('DIVERSE.XLM'!BrillouinDiversityCalc(B6:M6)),"")
	=IF(Total,SUM(B7:M7)/Total*EXP('DIVERSE.XLM'!DiversityCalc(B7:M7)),"")	=IF(SUM(B7:M7),SUM(B7:M7)/Total*EXP('DIVERSE.XLM'!BrillouinDiversityCalc(B7:M7)),"")
	=IF(Total,SUM(B8:M8)/Total*EXP('DIVERSE.XLM'!DiversityCalc(B8:M8)),"")	=IF(SUM(B8:M8),SUM(B8:M8)/Total*EXP('DIVERSE.XLM'!BrillouinDiversityCalc(B8:M8)),"")
	=IF(Total,SUM(B9:M9)/Total*EXP('DIVERSE.XLM'!DiversityCalc(B9:M9)),"")	=IF(SUM(B9:M9),SUM(B9:M9)/Total*EXP('DIVERSE.XLM'!BrillouinDiversityCalc(B9:M9)),"")
	=IF(Total,SUM(B10:M10)/Total*EXP('DIVERSE.XLM'!DiversityCalc(B10:M10)),"")	=IF(SUM(B10:M10),SUM(B10:M10)/Total*EXP('DIVERSE.XLM'!BrillouinDiversityCalc(B10:M10)),"")
	=IF(Total,SUM(B11:M11)/Total*EXP('DIVERSE.XLM'!DiversityCalc(B11:M11)),"")	=IF(SUM(B11:M11),SUM(B11:M11)/Total*EXP('DIVERSE.XLM'!BrillouinDiversityCalc(B11:M11)),"")
	=IF(Total,SUM(B12:M12)/Total*EXP('DIVERSE.XLM'!DiversityCalc(B12:M12)),"")	=IF(SUM(B12:M12),SUM(B12:M12)/Total*EXP('DIVERSE.XLM'!BrillouinDiversityCalc(B12:M12)),"")
	=IF(Total,SUM(B13:M13)/Total*EXP('DIVERSE.XLM'!DiversityCalc(B13:M13)),"")	=IF(SUM(B13:M13),SUM(B13:M13)/Total*EXP('DIVERSE.XLM'!BrillouinDiversityCalc(B13:M13)),"")
	=IF(Total,SUM(B14:M14)/Total*EXP('DIVERSE.XLM'!DiversityCalc(B14:M14)),"")	=IF(SUM(B14:M14),SUM(B14:M14)/Total*EXP('DIVERSE.XLM'!BrillouinDiversityCalc(B14:M14)),"")
	=IF(Total,SUM(B15:M15)/Total*EXP('DIVERSE.XLM'!DiversityCalc(B15:M15)),"")	=IF(SUM(B15:M15),SUM(B15:M15)/Total*EXP('DIVERSE.XLM'!BrillouinDiversityCalc(B15:M15)),"")
	=IF(Total,SUM(B16:M16)/Total*EXP('DIVERSE.XLM'!DiversityCalc(B16:M16)),"")	=IF(SUM(B16:M16),SUM(B16:M16)/Total*EXP('DIVERSE.XLM'!BrillouinDiversityCalc(B16:M16)),"")

	A	B	C	D	E	F	G
15	Proportion	=IF(Total,SUM(B5:B16)/Total,"")	=IF(Total,SUM(C5:C16)/Total,"")	=IF(Total,SUM(D5:D16)/Total,"")	=IF(Total,SUM(E5:E16)/Total,"")	=IF(Total,SUM(F5:F16)/Total,"")	=IF(Total,SUM(G5:G16)/Total,"")
16	Interval Totals	=SUM(B5:B16)	=SUM(C5:C16)	=SUM(D5:D16)	=SUM(E5:E16)	=SUM(F5:F16)	=SUM(G5:G16)

	H	I	J	K	L	M
15	=IF(Total,SUM(H5:H16)/Total,"")	=IF(Total,SUM(I5:I16)/Total,"")	=IF(Total,SUM(J5:J16)/Total,"")	=IF(Total,SUM(K5:K16)/Total,"")	=IF(Total,SUM(L5:L16)/Total,"")	=IF(Total,SUM(M5:M16)/Total,"")
16	=SUM(H5:H16)	=SUM(I5:I16)	=SUM(J5:J16)	=SUM(K5:K16)	=SUM(L5:L16)	=SUM(M5:M16)

	P	Q	R

		Shannon	Brillouin
14	Mean State Interval Diversity	=SUM(N5:N16)	=SUM(O5:O16)
15	Equality Good Intervals	=IF(Total,EXP('D:\THESIS\APPENDIX\DIVERSE.XLM'!DiversityCalc(B18:M18)),"")	=IF(Total,EXP('D:\THESIS\APPENDIX\DIVERSE.XLM'!BrillouinDiversityCalc(B19:M19)),"")
16	Temporal State Diversity	=IF(Total,EGI/MSTD,"")	=IF(Total,EGI_B/MSTD_B,"")
17	Temporal Evenness	=Hprime/LN(COUNT(Intervals))	=IF(Total<COUNT(Intervals),"",H/Hmax)

5	Total	=SUM(B5:M16)
6	Diversity	='DIVERSE.XLM'!DiversityCalc(dataArea)
7	Index	=EXP(Q6)
8	Hmax	=IF(Total<COUNT(Intervals),"",1/Total*LN(FACT(Total)/(FACT(X)^(s-r_)*FACT(Y)^r_)))
9	H'	='DIVERSE.XLM'!DiversityCalc(B18:M18)
10	X	=INT(Total/s)
11	Y	=INT(Total/s)+1
12	s	=COUNT(Intervals)
13	r	=MOD(Total,COUNT(Intervals))
14	H	=IF(Total,'DIVERSE.XLM'!BrillouinDiversityCalc(B19:M19),"")

Vita

Jerry Jacobs

Personal Data :

Born : December 17, 1958

Education :

BA in Biology, University of California, San Diego (December 1980)
BA in Chemistry, University of California, San Diego (June 1981)
MS in Marine Biology, University of California, San Diego (June 1984)
PhD in Zoology, University of Washington (1996)

Publications :

Ow, D.W., Jacobs, J.D., and Howell, S.H. (1987). Functional regions of the cauliflower mosaic virus 35S RNA promoter determined by use of the firefly luciferase gene as a reporter of promoter activity. *Proc. Nat. Acad. Sci.* **84**:4870-4874.

Baughman, G.A., Jacobs, J.D., and Howell, S.H. (1988). Cauliflower mosaic virus gene VI produces symptomatic phenotype in transgenic tobacco plants. *Proc. Nat. Acad. Sci.* **85**:733-737.

Jacobs, J., Ludwig, J.R., Hildebrand, M., Kukel, A., Feng, T.-Y., Ord, R.W., and Volcani, B.E. (1992) Characterization of two circular plasmids from the marine diatom Cylindrotheca fusiformis.: plasmids hybridize to chloroplast and nuclear DNA. *Mol. Gen Genet.* **233**:302-310

Wingfield, J.C., and Jacobs, J., (In preparation). Diversitiy Indices of Life History Stages and their sub-stages: implications for endocrine control mechanisms.