To the Graduate Council:

We are submitting herewith a thesis written by Brian E. Tucker entitled "The Development and Implementation of a Control System for a Hybrid Electric Vehicle." We have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Masters of Science, with a major in Mechanical Engineering.

J. W. Hodgson, Major Professor

W. R. Hamel, Major Professor

I have read this dissertation

and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor and

Dean of the Graduate School

THE DEVELOPMENT AND IMPLEMENTATION OF A CONTROL SYSTEM FOR A HYBRID ELECTRIC VEHICLE

A Thesis Presented for the Master of Science Degree
The University of Tennessee

Brian E. Tucker
May 1997

to my parents: for believing in me.

to Sherrie: for believing with me.

to my Lord and Savior, Jesus Christ: for giving me the **one** thing to believe in—You.

ACKNOWLEDGEMENTS

No one person nor any work he does is solely his own. This work has been wrought through the efforts of many who do not deserve to go unnoticed. My family's urging has kept me on course. My wife Sherrie's daily support has meant more to this project than she will ever know; thank you for never accusing me of being married to the car though it may have seemed like it. My parents also played a big role in my support structure.

One of the largest debts of gratitude, which I could never repay, I owe to James Hatmaker. His programming expertise, voluntary sacrifice of sleep, and cheerful spirit made the dark days of April 1996 not only bearable but possible. Without you, James, there would be no car.

Thanks also go to Dr. Johnson for serving as a faithful committee member. Also to my co-advisors Dr. Hamel and Dr. Hodgson who have endured my almost daily visits to their offices for advice and direction. Be assured I would still be at the first stages without all the work you have put in.

Finally, all glory belongs to God. In Him, we all live, move, and have our existence (Acts 17:28). He is worthy of all praise.

Abstract

In this study a complete control system for a parallel hybrid electric vehicle is developed from a theoretical standpoint and experimentally proven to provide performance similar to a conventional automobile. The vehicle chosen for this research is the UT-HEV, a 1995 Dodge Neon which was converted into an electric assist parallel hybrid vehicle for competition in the 1995 Hybrid Electric Vehicle Challenge. The approach to control system design includes a treatment of basic hybrid vehicle issues as well as the detailed development of models for both the electric motor and the internal combustion engine. From these models, a vehicle simulation was created to analyze the response characteristics of different control strategies. The control system was implemented through an embedded microcomputer control system coupled to various system components. The control software was developed to implement control of electric assist, regenerative braking, engine-driven charging, and pure-electric operation. Finally, empirical testing determined the final choice of control strategies for the vehicle. These tests focused on the system response characteristics as well as the vehicle's energy efficiency. The final control system has been proven to provide vehicle performance and response similar to today's typical vehicles.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

α		angular acceleration, rad/s
ρ_{air}		air density
θ		throttle position or road slope, degrees
τ_s		motor time constant, s
ω		angular velocity, rad/s
A_f		vehicle frontal area, ft^2 or m^2
B		damping coefficient
C_D		vehicle coefficient of drag
C_r		vehicle coefficient of rolling resistance
F		force, lbf or N
	F_{acc}	vehicle acceleration force
	F_{aero}	vehicle aerodynamic drag
	F_{drive}	total required vehicle drive force
	F_G	gravity force opposing vehicle motion
	F_{wheel}	vehicle rolling resistance force
g		acceleration due to gravity. ft/s^2 or m/s^2
i		current, A
	i_{des}	desired motor current
	i_{err}	error motor current

	i_m	actual motor current
J		polar moment of inertia
K_E		motor back-emf constant, V/krpm
K_{iv}		motor current feedback gain, %/A
K_T		motor torque constant, in-lb/A
L_{in}		PWM control voltage
m		vehicle mass, lbm or kg
m_i		equivalent vehicle inertial mass, lbm or kg
$\dot{m}$		mass flow rate, lbm/s or kg/s
	$\dot{m}_a$	mass flow of air into intake manifold
	$\dot{m}_c$	mass flow of charge out of manifold
N, n		rotational speed, rev/min
P		power, hp or W, or motor poles
	P_{drive}	total required vehicle drive power
	P_{EM}	electric motor power
	P_{HPU}	hybrid power unit power
p_m		manifold air pressure, kPa
R		universal gas constant or gear ratio
	R_g	transmission final drive ratio
	R_{EM}	electric motor pulley ratio
T		temperature, degrees Celsius

V		vehicle speed, miles/h, or motor speed signal, V or %
	V_{act}	"actual speed" motor current/speed feedback signal
	V_{auto}	actual vehicle speed
	V_{com}	motor speed control input
	V_{des}	desired vehicle speed
V_m		manifold volume, cm^3
Z_m		motor impedence, Ω

Abbreviations

CNG:	compressed natural gas
EM:	electric motor
EMC:	electric motor controller
EPROM:	electrically programmable read-only memory
ESS:	energy storage system (battery pack)
EV:	electric vehicle
HEV:	hybrid electric vehicle
HPU:	hybrid power unit
HVAC:	heating, ventilation, and air conditioning
ICE:	internal combustion engine
MAP:	manifold air pressure
NAAQS:	National Ambient Air Quality Standards
PHEV:	parallel hybrid electric vehicle

PNGV:	Partnership for a New Generation of Vehicles
PWM:	pulse-width modulation
RCS:	Real-Time Control System
SHEV:	series hybrid electric vehicle
SOC:	(battery) state of charge
TPS:	throttle position signal or sensor
ZEV:	zero-emissions vehicle

Chapter 1

RESEARCH OBJECTIVES

Overall

This study involves the design and implementation of a working control system for a hybrid electric vehicle. A heat engine hybrid electric vehicle uses both a heat engine taken to be an internal combustion engine in this study and an electric motor to supply energy which moves the car. The control system from this hybrid vehicle controls the interaction between these two energy sources.

The University of Tennessee's winning entry at the 1995 Hybrid Electric Vehicle Challenge (a student competition sponsored by Chrysler Corporation and the United States Department of Energy), a converted 1995 Dodge Neon, served as the test vehicle in this study. Though the vehicle won the competition, the original control system was deemed unsatisfactory for two reasons: first, it did not implement several areas of functionality desired for this vehicle. Second, the original control system proved difficult to reconfigure and thus was considered less than ideal for future research and competitions. Therefore, in the fall of 1995, the need for a new control system prompted this study.

Even though the test vehicle could function with its current control system, the range of this project led to the decision that the initial control system would be completely replaced. Therefore, the total scope of the research included specification, purchasing, and installation of new hardware as well as the configuration of the software needed to implement appropriate control strategies.

These control strategies would be determined through theoretical and experimental study.

Objectives

To ensure expected results, the boundaries and focus areas of the project have been expressed by three major research objectives:

1. *The final control system should produce a vehicle with driving performance, familiarity, and ease of operation similar to today's conventional cars.* This objective tells of the scope of the project and that it applies to the entire vehicle and not just one area of control. Therefore, the research must be broad in its approach and comprehensive in its attention to each of the areas involved in total vehicle control. As stated, special attention must be given to the fact that a new driver should have to make little or no adjustments in driving style to operate the final vehicle. This objective is subject to the drivetrain capabilities of the car.

2. *The control system should be designed in such a manner as to allow quick, easy changes in control strategies for research evaluation purposes in this study as well as for future studies.* This study will involve comparison of control strategies (see objective 3); therefore, such flexibility is paramount in the system design. In this objective statement comes implicitly the fact that this study will not be the last dealing with this vehicle; this project's breadth prohibits exhaustive optimization and therefore relegates that duty to future studies. To support this future effort, the current study should produce a control system with flexibility sufficient to accommodate any foreseeable needs of such a project. Meeting this need also involves extensive documentation.

3. *The control system strategies and settings will be based on a brief theoretical study augmented by experimental comparisons.* The focus of this study is not to develop complex models of the vehicle and its components in order to understand the optimum controller configuration and strategies. Instead, a broad but approximate theoretical treatment of the entire vehicle, supported by experimental results, will provide the means necessary to produce a working vehicle which satisfies the first objective of the research (i.e., implementation, not optimization).

Later chapters will provide more detail to each of these objectives once sufficient background information has been covered. They do, as stated, act as a boundary—as well as a starting point and focus—of the whole of the research.

Chapter 2

BACKGROUND

The Motivations for Hybrid Electric Vehicles (HEV's)

The Challenges Faced by Today's Conventional Vehicle

While recent years have seen a great transformation of the basic automobile, the years ahead will bring even more significant changes to the automotive industry. The 1970's saw two great oil crises which drove the direction of the automobile, if only momentarily, towards increasing fuel economy within the constraints of cost and consumer preferences. Since that time lower gasoline prices and consumers' desire for larger, less fuel-efficient vehicles (such as sport-utility vehicles) seem to have shifted the industry's focus further away from this goal. The 1960's brought to legislation the first restrictions on automotive emissions which by the 1980's had been met through the invention of the catalytic converter and more sophisticated, fuel-injection engine control systems. In each case in the past, modifications to the classic internal combustion engine powertrain provided the answer to the automobile's large problems.

However, times have continued to change. No longer is the 26.5 mpg corporate average fuel economy good enough, nor are current emissions standards strict enough to meet the needs of the environment. Today, over 1/3 of the United States trade deficit can be attributed to foreign oil dependence (the largest single contributor), and motor vehicles account for 63% percent of all of the oil used in this country [1]. From the emissions standpoint, California has seen the greatest air quality problems. According to the National Ambient Air Quality Standard (NAAQS) laid out by the 1990 Clean Air Act, Los Angeles currently exceeds

ambient air standards for ozone, carbon monoxide (CO), particulates, and nitrogen dioxide. Estimates state that motor vehicles account for more than half of the ozone precursor and CO emissions in California [2]. California is not the only hard-hit area: all Northeast states, except Vermont, currently violate the NAAQS for ozone and have at least one CO non-attainment area [2].

To promote development of, among other things, more fuel efficient vehicles, President Clinton and Vice President Gore joined forces with the chief executive officers of the Big Three automakers to form the Partnership for a New Generation of Vehicles (PNGV). On September 29th in 1993, the group pledged itself to the pursuit of three primary goals: To significantly improve national competitiveness in manufacturing, to implement commercially viable innovations in motor vehicles, and to develop a vehicle which would achieve three times the fuel economy of today's mid-range family car with no degradation in performance or cargo capacity. This program's plans to produce concept vehicles by 2000 and pre-production prototypes by 2004 has focused much immediate attention on new and innovative ways to meet this goal whose aspirations bring to mind the equally ambitious "man-on-the-moon" Apollo project [3].

The solution to the pollution problem has not been a simple one; one of the few alternatives is to make drastic reductions in pollutants in the face of increasing numbers of automobiles (and increasing vehicle miles traveled) is to introduce many vehicles which have essentially no emissions. The California Air Resources Board (who along with the EPA under the Clean Air Act is the only body able to legislate automotive emissions requirements) laid out a strategy to attack the severe emissions dilemma which included requiring that 2 percent of all light-duty motor vehicles sold in the state of California in 1998 must be zero emissions vehicles (ZEV's), i.e., electric cars [4]. This percentage was to increase to 5% in 2001, 10% in 2003, and 70% in 2010. Though this requirement was delayed,

automakers are already bringing electric vehicles to market such as GM's EV-1 which was first leased through Saturn dealers in California and Arizona in the fall of 1996.

The Proposed Solutions and Their Drawbacks

In response to the challenges ahead in both fuel economy and emissions requirements, conventional technologies which have held up the automobile through its first 100 years no longer seem to be up to the task. The PNGV program plan states:

> *The design space (for achieving the goal of three times fuel economy) has both theoretical and practical limits. On the basis of practically achievable thermal efficiencies with various heat engines, three times fuel economy may not be reached by engine improvements alone. The thermal efficiency needed ranges from approximately 40% to 55% which is about twice as efficient as today's engines. Even with advanced fuel cells, which have higher potential efficiencies than heat engines, other vehicle improvements are likely to be necessary [3].*

In addition, the emissions problem, as previously stated, cannot be met through incremental improvements; the severity of the problem requires a step-change of at least an order of magnitude to significantly impact California's potential air crisis. Even though some alternative fuels (such as natural gas) as well as some gasoline vehicles have shown great improvements, these changes alone will not be enough to make a difference.

For some, the electric vehicle (EV) seems to be the solution to all of the problems caused by the modern internal combustion engine automobile. The supporters of EV's purport that they are powered by the relatively efficient electric power grid

and that they produce no tailpipe emissions. Not only this, but an electric powertrain also has the ability to capture some of the vehicle's kinetic energy which is normally lost through friction in braking. Additionally, an EV does not incur "idling losses" associated with a normal ICE (internal combustion engine) car. In fact, during the 1995 American Tour de Sol, an electric vehicle proved itself twice as energy-efficient as an identical gasoline-powered "control" vehicle which traveled the same course [5]. It would seem that the electric car, then, provides a solution both to the fuel economy and emissions problems plaguing today's ICE-driving society.

With all the electric cars obvious advantages, no one would debate their inability to compete with the conventional automobile in many key areas. In the way of performance, all current electric vehicles suffer from severely limited range capability. The GM EV-1, the first fully-electric production car offered by a major auto manufacturer since near the turn of the century, has a projected highway range of only 90 miles; city mileage drops to an abysmal 70 miles, and using accessories like air conditioning, lights, and radios will reduce the range even further [6]. Ford's Ecostar and Chrysler's Epic minivan produce similar results [7] [8]. Even the reigning EV range champ, the Solectria Sunrise, has only mustered 238 miles on a single charge by 1995 [5], far less than most gasoline-powered vehicles. Recharging cycles on electric vehicles vary, though most require 3-8 hours to reach a full charge.

Price, another important point of comparison, shows the modern electric vehicle lacking in providing the value which today's consumer expects. The added complexities of an EV along with the battery pack combine for total vehicle prices two to four times higher than their conventional counterparts. The EV-1, which in performance could be compared to General Motors' Saturn Coupe, has a base price twice as high and offers less passenger room than its equivalent

Saturn. Another important consideration is the life cycle of the battery pack. Though it is the single most expensive component in most EV's, the batteries are also the least durable. Most battery packs would have to replaced every few thousand miles at a cost of several thousand dollars. No wonder, then, that GM offers the car only on a lease basis to consumers to avoid issues of maintenance costs.

In addition, while electric vehicles may be designed to decrease emissions, some cases might increase pollution problems (expecially in areas where coal is the main fuel for powerplants)[8]. This coupled with all the many factors listed above demonstrates that while the electric vehicle embraces sound technological concepts, its promulgation into a consumer-grade product has not yet been demonstrated.

While neither the improved ICE-auto nor the EV of today seem fit to play the part of savior to the American society, the exigencies which spawned their consideration have not abated. An intriguing solution, however, might exist. What if the positive aspects of both approaches could be married to a single design which embraced the time-tested and user-friendly internal combustion engine as well as the efficient, forward-thinking electric powertrain?

The Hybrid Electric Vehicle Alternative

Such a vehicle does not exist only in some engineer's hazy fantasy: hybrid electric vehicles, or HEV's, came into existence only a few short years after the automobile itself. The ability of an HEV to provide conventional vehicle performance while gaining electric powertrain efficiency could debunk traditional wisdom's "you can't have it all" attitude. Where the two previously mentioned technologies that can neither meet today's needs effectively, the hybrid electric vehicle could provide the range and performance of today's conventional cars

while still obtaining electric vehicle efficiency. Coupling this with the potential for low emissions (and even no emissions for periods of time), the HEV shows promise in ways neither of the other technologies can. In fact, the PNGV program research has centered around HEVs while California regulations may be changed to allow hybrids to met the zero emissions vehicle requirement.

Hybrid electric vehicles do have the ability to meet today's needs better than EVs or ICE vehicles. For example, the energy efficiency of a hybrid vehicle can double the fuel efficiency of today's vehicle [9], while their more conventional performance makes them attractive to a much larger audience than the electric vehicle. A study conducted in the Federal Republic of Germany in 1981 indicated that consumer acceptance of the electric vehicle due to performance restrictions would limit its acceptance to about 5.6% of the current market. Interestingly, because hybrids have no such limitations to their acceptance, their more widespread use would lead to a conservative estimate of 21.4% fuel savings versus the EV's 5.6% [10]. This advantage of acceptance produces obvious dividends. From the emissions standpoint, hybrids can also be operated as electric vehicles (ZEVs) for periods of time. All these factors clearly demonstrate the motivations for hybrid electric vehicles despite the prospect of (currently) higher costs.

Hybrid Electric Vehicle Concepts

Hybrid Definitions

As previously stated, hybrids derive their unique configuration from the use of two main power sources: an internal combustion engine and some form of electric powertrain. To further formalize the definition of such a vehicle, the Technical Committee 69 (Electric Vehicles) of the International Electrotechnical Commission set forth the following explanation:

Definition 2.1: "A hybrid road vehicle is one in which propulsion energy, during specified operational missions, is available from two or more kinds or types of energy stores, sources, or converters. At least one store or converter must be on-board.

"A hybrid electric vehicle (HEV) is a hybrid vehicle in which at least one of the energy stores, sources, or converters can deliver electric energy [4]."

An electric motor and/or generator fulfills the role of delivering electric energy while the other energy converter is typically an internal combustion engine.

Several primary components must be present in every hybrid electric vehicle. These include:

- **Electric Motor (EM):** In any HEV, an electric motor is mechanically coupled to the drivetrain of the vehicle. A motor controller usually handles any necessary inputs and power conversion and is for simplicity's sake considered a part of the electric motor system here.

- **Hybrid Power Unit (HPU):** Because all hybrids considered here will use an internal combustion engine as their second energy converter, this unit is sometimes called the "engine" or auxiliary power unit (APU). The HPU may or may not be directly coupled to the drivetrain and can provide power either to the wheels or, through a generator, to the energy storage system or electric motor.

- **Generator (GEN):** Used in some hybrids, this energy converter can take mechanical energy from the HPU and generate electrical energy to be used by the EM system.

- **Energy Storage System (ESS):** The energy storage system provides power to the electric motor system. Batteries fulfill the ESS need in most HEVs; however, flywheels, ultracapacitors, or any combination of the above have been tested.

- **Fuel:** This energy store provides the power supply for the HPU. Hybrids often use alternative fuels such as compressed natural gas (CNG) or methanol.

The arrangement of these components and thus the scheme of power distribution defines the type of hybrid electric vehicle. A series hybrid electric vehicle takes the power from the HPU/fuel converter/store pair and channels it through a generator to the electric motor and/or energy storage system. Therefore, the flow of power travels linearly from one component to the next with only the EM providing tractive power. In a parallel hybrid electric vehicle, each of the converter/store pairs works together to propel the vehicle. Two separate energy paths exist; their combination at the point of the main vehicle drive differentiates this concept from the series.

Configuration does not, however, define the entire power distribution range for HEVs. The controls system and strategies used in any hybrid may have a profound effect upon vehicle performance; in fact, two hybrids with the same components and arrangement may behave incredibly differently just due to differences in control strategies. More issues regarding HEV control will be discussed later.

Hybrid Benefits

The performance potential of HEVs has made this concept attractive in the face of increasing demands on automobile performance. Energy efficiency ranks

highest in this hierarchy; several studies, both theoretical and experimental, have shown hybrids' ability to stand out in this area. The typical ICE vehicle appropriates only about 16% of its total available fuel energy for actually moving the vehicle [9]. Engine losses, as well as idling losses, use up 76% of the total fuel energy in the urban driving cycle. An HEV, on the other hand, has several advantages. First, because of the electric powertrain, the ICE can be significantly smaller and therefore produce smaller engine losses. Also, the idling losses can be eliminated since some hybrids have no direct connection of the HPU to the wheels. Regenerative braking can actually recover some energy back into the system. These theoretical savings can total as high as a 100% gain in efficiency (using one-half the fuel of a conventional vehicle). Examples of this type efficiency have been demonstrated as well. Volkswagen's hybrid Golf uses only one-third the fuel of its ICE-only counterpart [10].

Because of their use of an electric power converter in the vehicle powertrain, HEVs have the ability to recapture vehicle kinetic energy as electric energy while braking just like an EV. This regenerative braking plays a key role in HEV energy efficiency; the PNGV program plan states that "an efficient regenerative braking system must be implemented in order to convert energy effectively [3]." Regenerative braking becomes possible due to the physical phenomenon of moving an electric conductor through a magnetic field which produces current which can be used to recharge the ESS (the reverse situation is what drives the vehicle).

Yet another dividend of the electric half of the powertrain is the ability to operate as a zero-emissions vehicle for a period of time. This feature is particularly attractive in pollutant-plagued urban areas in which hybrids could switch to a zero-emissions vehicle (ZEV) mode to reduce city emissions. The consumer,

however, does not pay the same range penalty which is necessarily a part of today's EV.

As previously mentioned, the hybrid electric vehicle is not a new concept. Patents for HEVs were issued as early as 1905 while a commercial HEV hit the market in the 1910s [4]. For a more complete history of hybrid electric vehicles, the reader is encouraged to read [4] and [11].

Hybrid Electric Vehicle Taxonomy

The idea of the hybrid electric vehicle goes far beyond the nebulous combination of the aforementioned components. As previously mentioned, two categories of hybrid electric vehicle exist: series and parallel. Power distribution drives the differences between the two; each operates on a fundamentally different philosophy on how to best deliver power to drive the wheels. The series hybrid follows the opinion that all drive power should be supplied by one component: the electric motor. This relegates the remainder of the components to a support role in serially providing power to this main driver. The HPU converts fuel energy into mechanical energy which, in turn, drives a generator to produce electrical energy to supply the ESS and thus the EM (in general, the generator output can drive either the ESS or the EM directly). The parallel HEV philosophy embraces the use of two separate power converters which have the ability to drive the vehicle. Therefore, two separate power flows—one from the ESS to the EM and the other from fuel to HPU—produce the total power output needed to drive the vehicle. While some vehicles attempt to combine both methods of power distribution into one powertrain [12], these strategies represent a small minority and will not be covered here.

Power distribution fills up only part of the motivation picture. In general, the missions of each of the two types of hybrids differ. Series hybrids most closely

resemble EVs in their configuration. Their HPUs tend to be smaller and, as such, their mission profile is often deemed "range-extender," i.e., using the HPU to extend the range of what would otherwise be an electric vehicle. The parallel vehicle, on the other, more closely resembles its ICE-powered cousins in its use of the HPU. One common parallel HEV mission, called "power-assist," uses the HPU as the primary source of motive power while the EM supplies assist for high power-demand situations. These concepts are further illustrated in Figure 1.

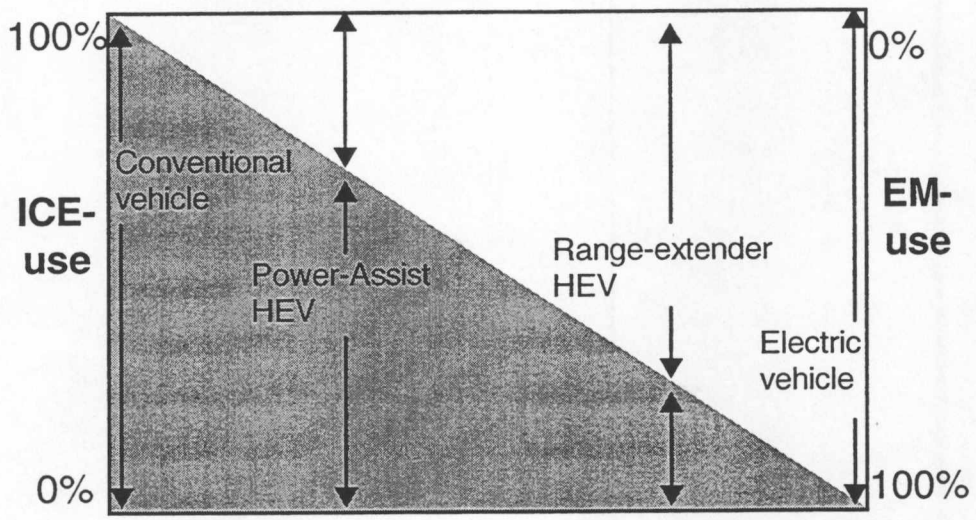


Figure 1: Typical HEV Mission Profiles

Figure 1 demonstrates the general trend of parallel hybrids (represented here by the power-assist HEV) to lean towards greater ICE use and the series hybrids tendency to more closely follow suit with their EV cousins.

Series Hybrid Electric Vehicles

The Series Hybrid Concept

The definition of a series hybrid electric vehicle (SHEV) can be derived directly from the previously mentioned power-distribution concepts:

> *Definition 2.2: "A series hybrid is an HEV in which only one energy converter can provide propulsion power."—Technical Committee 69 (Electric Road Vehicles) of the International Electrotechnical Commission [4]*

Figure 2 shows the arrangement of the basic HEV components in a series hybrid.

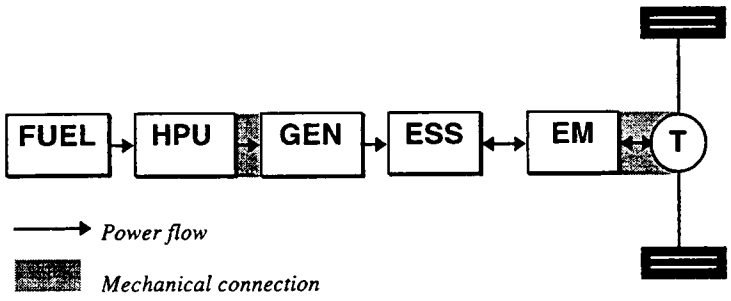


Figure 2: Series Hybrid Configuration

As stated in the SHEV definition, only the electric motor can supply power to drive the wheels. The implication of this statement, then, is that no direct mechanical connection exists between the HPU and the wheels. Also, as the name implies, the linear power distribution scheme passes power from one component to another in serial fashion.

One typical example of the series configuration can be found in the University of Tennessee "High Potential" SHEV which competed in the 1993 and 1994 Hybrid Electric Vehicle Challenge Ground-Up class [13]. A 32-kW electric motor provided tractive power to the rear wheels through a standard 5-speed manual transmission. A pack of fifteen 12-volt lead-acid batteries supplied the necessary energy storage (ESS) while a small gasoline-powered internal combustion engine drove a small generator set to supply additional power. This extra power made possible the recharging of the battery pack when necessary, providing further range to what would have otherwise been a pure electric vehicle.

Series Hybrid Benefits and Liabilities

Simplicity remains one of the largest draws of the series configuration. The SHEV represents an EV with the addition of the HPU and associated peripherals. Since only the EM must be mechanically connected to the wheels, the remaining components may be placed in any convenient location. Interconnection of components represents a minimum of complexity as well (see Figure 2). In addition, while Figure 2 shows the EM connected to some sort of mechanical transmission "T," such a transmission may not be needed in some series hybrid; this also the case in many EVs (the EV-1 is an example).

The HPU-generator set does not have any mechanical limits and can therefore run at any speed-load condition. In general, the HPU's size can be chosen such that it provides the power required to maintain ESS energy storage at the optimum engine operating condition. Throwing off mechanical constraints also means that idling losses need not occur; the HPU must operate only when called upon to do so (generally under ESS low state of charge conditions). Lower emissions are then inevitable result of both of these improvements (particularly if an electrically heated catalyst is used to reduce emissions when the HPU is started).

The series vehicle does retain a list of restrictions. Due to the losses in the power distribution chain, recharging of the ESS through the use of the HPU tends to promote fairly low efficiency energy conversion. This becomes even more apparent when compared to charging the ESS directly from the electric power grid. Therefore, to obtain maximum efficiency, the HPU-generator set should be sized to provide only a small increase in range. In this "range-extender" example, most of the power could still come from the electric power grid and losses could be minimized [10].

Parallel Hybrid Electric Vehicles

The Parallel Hybrid Approach

Parallel hybrids, as previously mentioned, use their energy stores and converters in a much different way than the series hybrid:

> *Definition 2.3: "A parallel hybrid is an HEV in which more than one energy converter can provide propulsion power." –Technical Committee 69 (Electric Road Vehicles) of the International Electrotechnical Commission [4].*

Figure 3 shows the typical component arrangement in a parallel hybrid.

Figure 3 illustrates the power flow and mechanical connections of the parallel hybrid electric vehicle (PHEV). As stated in the definition, either the EM or the HPU or both have the ability to provide power to drive the vehicle. In general, one of these driving components acts to "assist" the other; in this way, neither one need be sized for maximum power requirements. Also apparent from the figure is the lack of the generator which was required in the SHEV; because of the mechanical connection between EM and HPU (which is the case in most

PHEVs), the HPU can drive the EM as a generator directly (assuming the EM can be so operated). The figure clearly demonstrates the appropriate choice of the name "parallel" for this type hybrid.

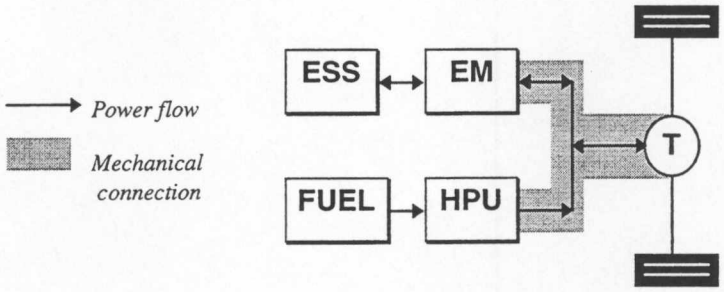


Figure 3: Parallel Hybrid Configuration

A large variety of options besides the one shown in Figure 3 still fit the PHEV definition. Three categories of PHEVs exist: torque-combining, traction combining, speed combining [10]. Torque combining hybrids have a direct mechanical connection between the HPU and the EM. In this way, the torque outputs of the two driving components may be added mechanically before reaching the final drive (or subtracted in the case of charging). Most PHEVs employ gears, belts, or transmission roller chain to achieve this task. This arrangement, as shown in Figure 3, requires the use of two shafts and takes the name dual shaft or offset torque-combining parallel hybrid electric vehicle. Alternatively, the HPU and EM could also be driven on the same shaft as in the inline torque-combining configuration. Traction combining hybrids bring the point of power addition outside the vehicle and on to the road. In this case, each main power converter drives one set of wheels. Therefore, the power provided by

either component or both can propel the vehicle. Finally, the speed-combining PHEV places a differential gearbox between the EM and HPU and attempts to combine the rotational speeds of each component. However, this arrangement requires that each component provide the same amount of torque to the differential gearbox. Speed-combining PHEVs are rare but possible. The remainder of this study will focus on the torque-combining PHEV; therefore, future references to parallel hybrids insinuate the torque-combining variety (i.e., mechanical connection between HPU and EM; HPU can drive EM as generator; no separate generator required).

The UT-HEV Neon, which is the subject of this study, provides an excellent example of an offset torque-combining parallel hybrid. A 32-kW electric motor receives electric energy from a 196V lead-acid battery pack; a natural gas powered 1.0L 3-cylinder automotive engine serves as the HPU. These drive a jack shaft attached to the transmission through a toothed belt. The HPU provides the primary drive power while the EM provides electric assist during high load conditions. More details about this vehicle will follow.

The Parallel Hybrid's Attractions and Distractions

Parallel hybrids have their own advantages: because of the mechanical connection between the EM and HPU, a separate generator may not be required. This therefore lessens the cost and complexity of the powertrain. Also, since both power converters have the ability to propel the vehicle in parallel, neither need be sized for the peak power requirements of acceleration or steep grades. This enables the PHEV designer to size one component to provide maximum efficiency under cruising conditions while using the other to help shoulder the heavier loads. Also, because each component may directly drive the vehicle, the PHEV drivetrain incurs low energy conversion losses.

This configuration is not without disadvantages. As might be expected, the powertrain layout itself lies at the root of most of these liabilities. First of all, the interconnection of components from the mechanical standpoint proves complex at best. Additional mechanical complexity could lead to reliability issues. Component interconnection presents complexities in trying to control power flow throughout the system. What decides when the electric motor would provide "assist"? When should the HPU drive the EM in generator mode? These questions are not easily answered. In addition, the mechanical connection of the HPU to the drivetrain can have a negative impact on emissions: the engine speed, now directly linked to vehicle speed through the transmission, can change rapidly possibly producing high emissions [11].

Hybrid Comparisons: Series Versus Parallel

Parallel and Series Hybrids - Two Approaches, One Mission

Today's hybrid electric vehicle research maintains a split focus: both parallel and series HEVs have been built and tested. With the assets and liabilities of each design presented, what motivates the choice of one type of hybrid over the other?

As previously mentioned and shown in Figure 1, the missions of each type of HEV seem fundamentally different. The SHEV, with its focus on the electric side of the powertrain, has the ability to provide a range-extended EV. The power-assist PHEV, on the other hand, promotes greater and more conventional use of the internal combustion engine. However, only one true mission exists: to provide efficient transportation for the world's consumers. Several factors including energy efficiency, emissions, packaging, and cost will be considered.

Making the Choice - Energy Efficiency

Energy efficiency depends largely upon component arrangement and each configuration receives certain characteristics based on this choice. To provide

conventional range and performance in the SHEV, each component must be sized to meet peak power requirements [10]. Obviously, since only the EM drives the wheels, it must be capable of providing the maximum required sustained vehicle power, P_{max}. This power may be required over an extended period of time in high-speed driving or climbing long hills. In either of these cases, should the batteries become depleted, both the generator and the HPU must provide P_{max} to the EM to sustain vehicle performance. Therefore, the total required power capacity of the components is 3 P_{max}. In contrast, the PHEV uses two components to propel the vehicle. In the worst case where both components were sized to provide P_{max}, then the total power capacity of the components would be 2 P_{max}. Either case cited represents an upper-limit: in practice, the ESS would be sized to provide some reasonable buffer of energy capacity allowing at least some of the remaining components to be sized at less than peak power capacity. However, the intent to demonstrate the PHEVs advantage in this area still succeeds in ideal or non-deal situations.

Another issue of energy efficiency related to hybrid configuration choice revolves around the issue of using the HPU to recharge the ESS. This feature is necessarily a part of the SHEV; the PHEV may or may not choose to employ it. Due to the chain of energy conversion components, an associated chain of efficiencies is present. Figure 4 demonstrates the differences in using the HPU to provide energy to the ESS compared to charging from the electric power grid.

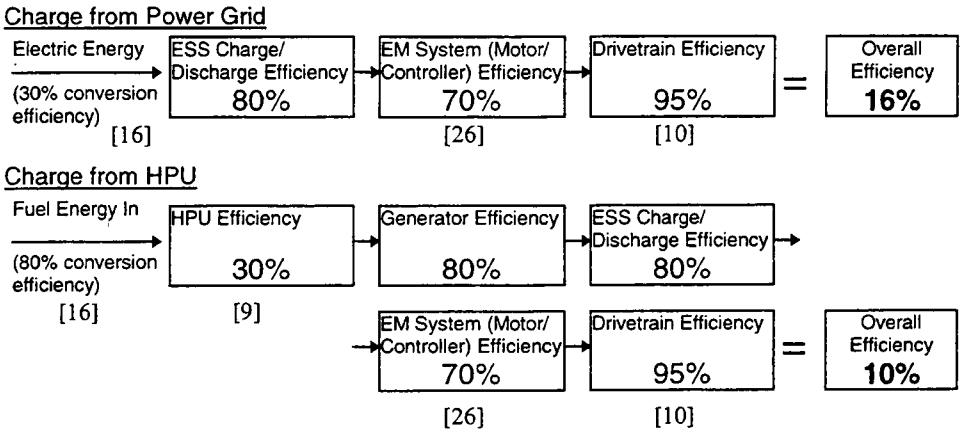


Figure 4: Energy Conversion Efficiency Comparisons

All values in Figure 4 are referenced with the exception of ESS and generator efficiency (assumed to be 80%). In Figure 4, the "conversion efficiency" term associated with charging from the power grid includes conversion of oil to electricity as well as efficiency of power distribution. The efficiency of fuel energy conversion is associated with energy lost in refinement of crude oil as well as in transportation. Obviously, the ability to use on-board charging may not be as advantageous as originally believed. Because this type of charging is the cornerstone of the SHEV (in fact, this is the only feature which differentiates it from an EV), its efficiency can suffer greatly. A small HPU/generator could reduce these losses by supplying less total energy to the ESS; however, supplying less total energy would require either a larger ESS or smaller vehicle range. This is why the "range-extender" concept appears to be the SHEV's most logical mode of application: HPU size is efficiency limited. Only in the PHEV can this unfavorable charging option be avoided.

Other Factors Affecting the Choice - Emissions, Packaging, Complexity

The emissions advantage fall with the SHEV. Because of its ability to operate its HPU under constant conditions, emissions are always predictable and can be

more easily managed than in the speed-dependent conditions prevalent in the PHEV.

Size, packaging, and cost play an important part in design motivations as well. While the PHEV has few components, the SHEV maintains more flexibility as to component placement; whereas the parallel's HPU and EM must be mechanically coupled, they must be located together. This requirement does not apply to a series vehicle. The parallel does sport one more size advantage, however: because of its lower use of the electric powertrain (versus the series), its ESS requirements tend to be smaller producing proportionate changes in size and cost as well. The lower component count also swings the cost advantage to the parallel hybrid.

The importance of controls and complexity cannot be overlooked. The parallel vehicle requires a more complex mechanical arrangement and controls system. This may negate some of the components cost gains afforded versus the series. Also, the added complexity brings with it issues concerning reliability.

Summary of Choices

In conclusion, the series hybrid is heavier, larger, and more expensive. It is also less complex, more reliable, and could potentially produce lower emissions [11]. The parallel HEV weighs less and costs less, produces higher energy efficiency under most conditions, and provides the greatest overall flexibility. Therefore, no clear-cut answer as to which is best under all conditions can be ascertained.

The remainder of this discussion will be directed toward the parallel hybrid configuration. Several factors motivated this decision and will be discussed along with the test vehicle description.

Parallel Hybrid Electric Vehicle Controls

As mentioned previously, controls define much of the performance of a hybrid electric vehicle and are especially important for a parallel HEV. The possibilities—and problems—seem endless. However, in the case of the PHEV, what must be controlled?

Figure 5 illustrates the point further. As shown, the key decisions involve when and how to use each of the main power converters.

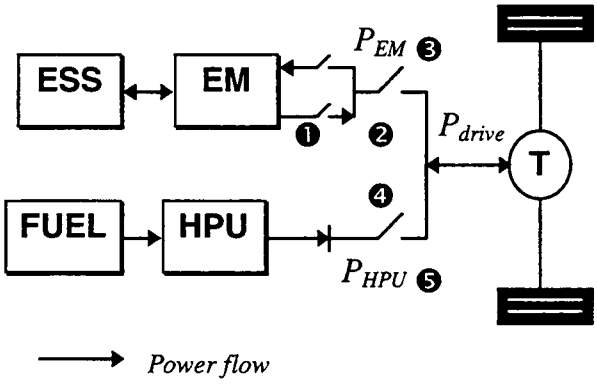


Figure 5: Parallel HEV Control Decisions

The numbers 1-5 in Figure 5 refer to each of the main powertrain control decisions. These are listed below:

1. **EM mode:** because the electric motor can act as either a motor (supplying power to drive the vehicle) or a generator (requiring power from the HPU to charge the ESS), any controller must control this decision.

2. **EM enable:** the EM may or may not contribute to the powertrain at any given time based upon vehicle needs and control strategies.

3. **EM Power:** should the EM be enabled, the level to which it contributes power to the drive system must also be controlled.

4. **HPU enable:** the HPU may or may not contribute to the powertrain at any given time based upon vehicle needs and control strategies.

5. **HPU power:** should the HPU be enabled, the level to which it contributes power to the drive system must also be controlled.

In any control approach, driver commands and physical constraints define the envelope within which the vehicle must operate. The total power supplied by the electric motor, P_{EM}, and the total power supplied by the HPU, P_{HPU}, must be at least equal to the required drive power such that

$$P_{EM} + P_{HPU} = P_{drive} \tag{2-1}$$

where P_{drive} is the drive power required. Because the aforementioned driver commands and physical constraints fluctuate, the answers to these five questions could very well be different each moment. Beyond meeting this constraint, the possibilities of control are open to any number of combinations of EM/HPU usage.

Several possible schemes exist; Table 2.1 shows all of the identified choice of operating modes for a PHEV [14].

In each case of driver inputs and physical constraints, one operating mode could prove more advantageous than another. Due to the nature of the design of the test vehicle, these modes will be discussed in detail later it their appropriate context.

Table 2-1: Parallel HEV Operating Modes

Mode	Description
Electric Mode	EM supplies all of drive power ($P_{EM} = P_{drive}$, $P_{HPU}=0$)
ICE Mode	HPU supplies all of drive power ($P_{HPU} = P_{drive}$, $P_{EM}=0$)
Primary Electric Mode	The EM provides principal drive power; augmented by HPU ($P_{EM}>P_{HPU}$)
Primary ICE Mode	The HPU provides principal drive power; augmented by EM ($P_{EM}<P_{HPU}$)
Hybrid Mode	HPU and EM split power to the drive in some way ($P_{EM}\approx P_{HPU}$)
ESS Charge Mode	HPU provides propulsion power and power to charge ESS with EM acting as generator ($P_{HPU}-P_{charge}=P_{drive}$)
Regenerative Braking	Braking energy recovered through EM acting as a generator and used to charge ESS ($P_{HPU}=0$; $P_{EM}<0$)

Several driver inputs may affect the choice of operating modes. These include:

- Accelerator position
- Brake
- Manual mode control

Physical constraints important in the consideration of operating modes and control decisions include:

- Vehicle physical properties (weight, aerodynamic drag, etc.)
- Environmental effects (grade, wind, etc.)
- Component constraints (ESS state of charge, EM/HPU output limitations)

Each of these individual categories will be dealt with in more detail in the coming chapters.

Though up to this point much has been said on *what* needs to be controlled, little has been said on *how* this can be accomplished. Unfortunately, though equation (2-1) drives any control system, no direct measurement of P_{drive} exists; this value must be estimated from system inputs. Therefore, it becomes the job of the System Controller to interpret driver inputs, external conditions, and internal conditions and determine the answers to the five powertrain control questions. The context diagram shown in Figure 6 schematically demonstrates this point.

Therefore, the Control System may be defined as all components necessary to carry out the task of powertrain control for the hybrid electric vehicle (i.e., answering the five powertrain control questions). The cornerstone of the control system is the System Controller; this component is defined as the unit which interprets the information from external and internal sources, decides from these inputs the necessary control outputs, and generates/coordinates carrying these commands out. The control system is the focus of this study.

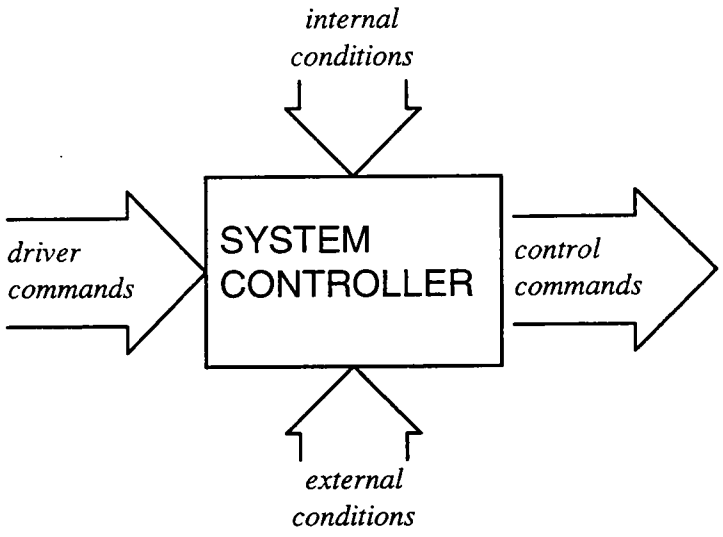


Figure 6: PHEV System Control Context Diagram

The 1996-1997 UT-HEV Neon

The vehicle used to carry out the research in this study is the 1996-1997 UT-HEV Neon. The UT-HEV began life as a 1995 Dodge Neon donated by Chrysler for use in the 1995 Hybrid Electric Vehicle Challenge in which it garnered first place. As part of the vehicle conversion, the design team chose to employ the parallel hybrid approach for several reasons:

> *"The series configuration required one large motor...but efficiency is low under cruising conditions. Also overall efficiencies could be compromised....*
>
> *With a parallel configuration the (vehicle simulation) software predicted that a medium sized auxiliary power unit and a medium sized electric motor could provide acceptable acceleration. [15]"*

The proposed competition rules further motivated this decision: the initial requirement of a 14 second 0-100 km/h acceleration seemed too steep for most EMs available. Therefore, to avoid the dilemma of a large, single traction motor, the design team chose an offset torque-combining parallel hybrid configuration. Eventually competition organizers dropped this steep acceleration requirement; however, this change came too late to alter the design.

Table 2.2 details the various components chosen for the UT-HEV:

Table 2.2: UT-HEV Components

Component	Description
EM	Permanent Magnet, Brushless DC Motor
	Unique Mobility SR-180P (motor), CR20-300-1 (controller)
	Rated Power: 31.3 kW (continuous), approx. 46.2 kW (intermittent)
ESS	Lead Acid Battery Pack
	Exide U1 DC-9 deep-cycle lead acid batteries (16)
	Voltage: 196 (nominal); Rated Capacity: 2.9 kWh; Weight: 150 kg
MPU	Four-stroke Internal Combustion Engine (liquid cooled)
	Geo (Suzuki) 1.0L, 3-cylinder
	Maximum Power: approx. 33 kW
Fuel	Compressed Natural Gas (CNG)
	Capacity: approx. 5.5 gallons (gasoline equivalent) at 25 MPa

This vehicle, once completed, garnered first place in the Neon class at the 1995 Hybrid Electric Vehicle Challenge. The vehicle's standout emissions

performance as well as consistency in every event contributed to this success. Figure 8 shows the vehicle in mid 1996.

The new challenge for the UT-HEV in 1996 consisted of the Hybrid Electric Vehicle Challenge coupled with the American Tour de Sol (ATdS) sponsored by the Northeast Sustainable Energy Association (NESEA), Chrysler Corporation, and Argonne National Laboratory. The focus of this competition centered mainly around energy economy and, indirectly, vehicle reliability.

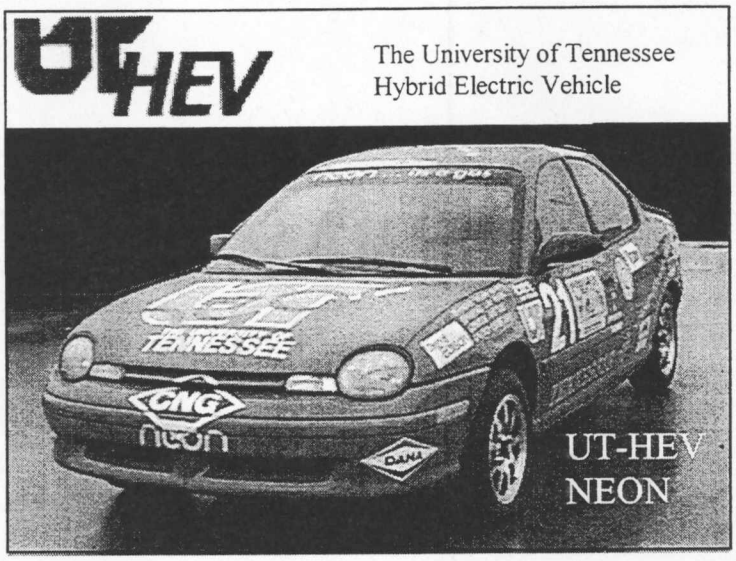


Figure 7: 1995 UT-HEV

Dynamometer testing would occur during the first week while the running of the ATdS, a 311 mile road rally, filled the second week of competition. Of the 1000 total available points in the competition, energy economy encompasses 400 points; the next largest single category, emissions, entails only 150 points.

Competition rules also provide restrictions regarding the vehicle controls system. Section C-2.1.7, "Driving Modes," recommends that both HEV and ZEV modes be available for each vehicle. Section C-2.2.4, "Passive Control Strategy," further states that the HEV mode must have a passive control strategy:

> *"Manual adjustments to devices directly controlling current, voltage, frequency, etc. to the traction motor or the engine system or drivetrain are not allowed, excluding conventional vehicle controls. If programmable control devices are used, teams must develop a single strategy and calibration flexible enough to operate the vehicle in all events [16]"*

All of these factors drove the design of the control system for the UT-HEV and its final control strategies.

Chapter 3

SYSTEM REQUIREMENTS AND THEORETICAL APPROACH

Approach Introduction

Chapter 2 outlined the basic concepts relating to the control of a parallel hybrid electric vehicle. The requirements development process and the requirements document follow first in this chapter; these elements define the approach to the problem solution. This chapter will also build on the theoretical ideas developed previously (the five control questions and the fundamental equation) and apply them to the UT-HEV. In addition, the internal conditions, external conditions, and driver commands will be clearly defined and quantified. Chapter 4 then deals with the hardware development and integration tracing the path of logic and software development through current lines of thinking and the approach chosen for the UT-HEV. Finally, Chapter 5 deals with the approach to system testing which leads into the final results and conclusions about the finished system.

Requirements Development

Any good software design project (of which this project is one) must follow a formalized progression of steps from the initial concept to the finished code. Formalizing a set of requirements for the end product is the first step towards a well-polished and organized design. This area of software design has received considerable attention in the past several years and has led to the development of standards regarding how these requirements should be written; IEEE standard 830 and DOD-STD-2167A are the foremost examples [17]. In each of these approaches, the requirements may contain very formal functional definitions

regarding system operation, throughput, etc.; however, the UT-HEV control system does not require the level of detail specified by these documents.

The author developed the requirements document to identify and partition each of the functional areas deemed necessary to meet the objectives laid out in Chapter 1. Additionally, the requirements define the terms used to refer to each subsystem and its operation. Finally, each functional area of operation is discussed relative to its required performance.

System Requirements

The complete requirements document, completed in January of 1996, follows.

The control system for the UT-HEV Neon must meet the following functional requirements:

Overall Requirements

The control system, which is comprised of the system controller, display, driver interface, sensors, wiring, and data storage, must give the driver choice of the vehicle operating modes (ZEV or HEV) and provide the control necessary to sustain each mode. The system may also be called upon to provide peripheral control of accessory functions such as HVAC as well as to provide additional functionality in the area of data acquisition.

In both modes, the system controller must provide speed and regeneration control of the electric motor. In the ZEV mode, the system controller provides the speed input necessary to drive the electric motor based upon driver directives such as through the accelerator pedal. In the HEV mode, the system controller must decide when to turn the electric

motor on and for what reason. Two cases exist: the motor can react in "assist" where the motor provides extra power to drive the vehicle. Also, the motor can "regenerate" to help re-charge the batteries, if needed. In both modes, the controller must oversee the function of regenerative braking. The system controller will also be responsible for monitoring the battery state of charge. In addition to these tasks, the system controller will provide, through the display, the display and monitoring functions required to keep the driver informed of the state of the vehicle at any time. This function also should include the continuous monitoring of vital component states (e.g. battery temperature).

Functional Requirements

The following represents, in detail, the specific requirements for each of the functional categories.

Mode Control: The system controller must provide for driver selection of the two operating modes of the vehicle (ZEV and HEV). Once selected, the system controller must maintain the control strategies associated with each until the vehicle shuts down (i.e., the key is turned off). The system controller is not required to change operating modes while operating; the system controller will remain in the state selection given upon start-up until system shut-down.

ZEV Motor Control: The control system must provide the means by which the electric motor will drive the vehicle in ZEV (electric motor only) mode. This includes taking the driver's throttle position as an input. The final speed of the motor and thus the vehicle should be proportional to the driver's commanded throttle position.

Electric Assist: The control system must provide the means to control the electric-assist function of the vehicle. Electric assist is defined as using the electric motor to add torque towards the goal of driving the vehicle. In "normal" operation, the electric motor (EM) is free-turning and the hybrid power unit (HPU) provides all the motive power. Assist adds extra power to drive the vehicle. The control system must decide when to assist with provision for manual driver dictation of assist mode. The system controller should also provide assist during vehicle take-off to ensure proper performance. The assist function is only active in HEV mode.

Regeneration: The system controller should provide the means to control the regenerative operation of the electric motor. Regeneration is defined as using the motor as a generator to provide power to the batteries. This can occur under three conditions: first, the control system must be able to implement a regenerative braking scheme in which energy which would normally be dissipated during braking is absorbed by the motor/generator and sent to the batteries. Second, if the state of charge (SOC) of the batteries is low, the controller should automatically use the HPU to drive the motor as a generator to bring the battery charge to an acceptable level. Third, driver commanded regeneration should generate the same result as would low SOC. The regeneration function is active in both modes, however, only regenerative braking is available in ZEV mode.

Battery State of Charge: The system controller will monitor the SOC of the battery pack. This data will then be used to support other processes

(such as regeneration). This may be accomplished through monitoring of the battery pack voltage, discharge current, and temperature.

Monitoring: The control system will also be responsible for monitoring the critical states of the various vehicle components as well as informational data. These include battery pack temperature, EM fan state (is the cooling fan on?), EM temperature, CNG tank pressure, CNG tank temperature, engine (MPU) speed, and vehicle speed.

Display: The system controller will also update the driver of the state of the vehicle at regular (at least 1 second) intervals. These outputs will be shown on the display hardware and may include battery SOC, temperatures, and engine/vehicle speeds. The display should be flexible to accommodate driver needs.

Driver Interaction: The system controller also has the task of interfacing with the driver's control inputs and passing these values on to the operational modules. These include mode select, "dictator" controls for manual assist and regeneration, and display controls.

Data Acquisition: Another function of the system controller is that of data acquisition. This function will take data fed to the system controller through the functions described above and store that information for future use by the development team. This storage should be configurable to fit the needs of the situation and the data should be stored either on a floppy disk or on the hard drive. Finally, the information must also be able to be sent across the network connections supplied with the system.

HVAC: The specific requirements of this function have yet to be defined. It will, however, run separately from the previous processes.

The sole change to these requirements relates to the HVAC (heating, ventilation, and air-conditioning) functional area; this requirement has been dropped altogether.

Controls Basics Revisited

Making Control Decisions

With the system requirements now laid out, the next step in the development approach is to again consider the basic concepts explained earlier and build from this point a theoretical basis with which to justify the control strategies to be implemented. First of all, recall the five control questions as discussed in Chapter 2 and repeated here in Figure 8.

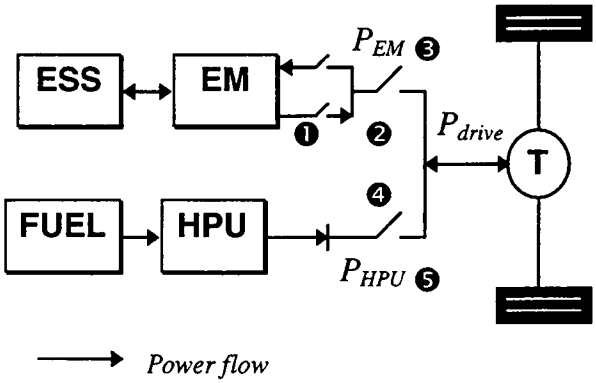


Figure 8: PHEV Controls Decisions - Review

Again, as shown in Figure 8, five control decisions must be made. For the EM, operating mode (motor or generator), enable, and power output decisions must

be made. Additionally, the HPU can be enabled or disabled; its power must be controlled as well. The approach of the UT-HEV to these decisions will be dealt with by looking at each of these two power converters in turn.

The first area of discussion is the HPU. As with any homogenous charge spark-ignition internal combustion engine, its control input is the position of the throttle; increasing the throttle opening will, in general, increase the power output of the HPU. In a typical vehicle, the driver has direct control over the internal combustion engine's throttle and thus its power output. A cable linking the accelerator pedal directly to the throttle plate manages this control. This mechanical control method is simple, safe, and effective. For these same reasons, this type of direct driver control was chosen for the HPU in the UT-HEV. In this way, possible dangers associated with "drive-by-wire" control (i.e., servo-actuated throttle control in which no direct mechanical connection exists between the accelerator pedal and the throttle) have been eliminated. Also, this method of control provides the driver with a feel much more similar to a normal automobile.

The driver also controls HPU enable/disable: through the selection of HEV mode, the driver enables the HPU; in ZEV mode, the electric motor alone propels the vehicle and the HPU is disabled.

The Role of the System Controller

Thus far, control has focused on the driver as the controlling element and the HPU as the system being controlled. However, the EM does not fall under direct human control; it is therefore the role of the system controller to react to driver inputs and the HPU's state to make the remaining control decisions.

The fundamental equation previously used (2-1) to describe the relationship between available and required power further demonstrates this important role.

Through the previous discussion, it has been established that the driver controls the power output of the HPU through the throttle command. Also, as will be discussed in more detail later, the driver takes in the external conditions as well as a pre-determined "desired" vehicle speed and thus controls, in a sense, the drive power required at the wheels. If, then, these two terms in (2-1) are not under the control of the system controller, we can rearrange the equation in terms of driver-determined parameters and the main control variable, i.e., the electric motor power output.

$$P_{EM} = P_{DRIVE} - P_{HPU} \tag{3-1}$$

Therefore, the system controller must sample and/or interpret the other two variables and determine the required motor power output. Included in this decision is the question of EM mode. Should the motor operate as a motor, $P_{EM} > 0$; as a generator, $P_{EM} < 0$. Finally, deciding to use the motor at all determines the solution to the enable/disable question.

As a result, the final control diagram is shown in Figure 9. Here, the driver and system controller take their proper place (for the UT-HEV) in answering the five control questions.

Modeling the Required Drive Power

The P_{drive} element of (2-1) depends upon two general factors: road load and driver commands. The exploration of each is necessary to understand the full control picture and to thus calculate the required power from the EM in (3-1).

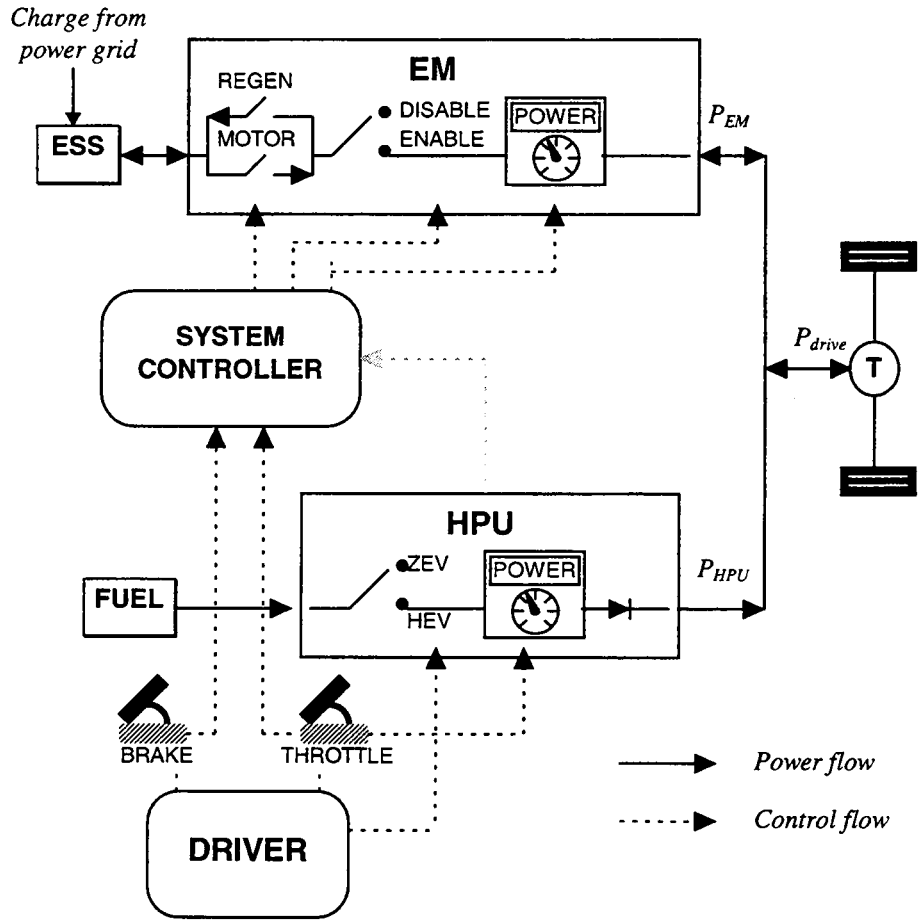


Figure 9: Vehicle Control Schematic

Road Load Modeling

External conditions which are independent of the driver relate vehicle speed to required drive power. Figure 10 shows forces which have a component that acts upon a vehicle in the direction of motion.

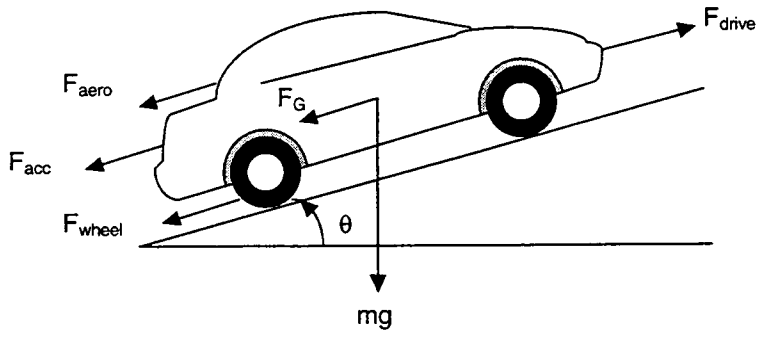


Figure 10: Forces Acting Upon Vehicle in Motion

Taking the summation of these forces in the direction of motion, the net required drive force can be expressed as

$$F_{DRIVE} = F_{AERO} + F_{WHEEL} + F_G + F_{ACC} \tag{3-2}$$

where these forces include aerodynamic drag (F_{AERO}), rolling resistance (F_{WHEEL}), gravitational force (F_G), and accelerative force (F_{ACC}).

The aerodynamic drag force, governed by the following equation, plays a key role in vehicle road load, especially at high speeds:

$$F_{AERO} = \frac{1}{2}\rho_{air} C_D A_f V^2 \tag{3-3}$$

where ρ_{air} is the density of air, C_d and A_f are the drag coefficient and frontal area of the vehicle, respectively, and V is the velocity of the vehicle in stagnant air. This expression, being a strong (2nd order) function of vehicle velocity, can dominate all other effects as the vehicle speed increases.

Vehicle rolling resistance is also important, and may be expressed as

$$F_{WHEEL} = (mg\cos\theta)C_r(V) \tag{3-4}$$

where C_r is the coefficient of rolling resistance or rolling friction; this coefficient varies with vehicle velocity [18, 19]. The rolling resistance is commonly taken to be of the form

$$C_r(V) = K_0 + K_1 V + K_2 V^2 \tag{3-5}$$

for rolling resistance coefficients defined as K_0, K_1, and K_2. This quadratic relation accounts well for the changes in the coefficient's value with speed [18]. The quadratic relation has also been used elsewhere [20]; however, some research has been conducted using a third-order representation of rolling resistance [21], while yet others [22] use a linear approach. This study will follow through with the quadratic approximation as shown in (3-5); this may be subject, however, to availability of these constants from the manufacturer.

When the vehicle is traveling up a grade of angle θ (as shown in Figure 10), it is then important to consider the component of the vehicle's weight which opposes its forward motion (or, if traveling down a grade, the force which aids the vehicle's motion). This force can be simply expressed as

$$F_G = mg\sin\theta \tag{3-6}$$

Finally, the power required for acceleration may be computed using Newton's Second Law; the resulting expression is

$$F_{ACC} = m_i a \tag{3-7}$$

where a is the required acceleration and m_i is the equivalent inertial mass. It is important to understand that this value of inertial mass will be greater than the actual vehicle mass. This effect results from the need to accelerate the angular velocity of all the rotating components of the vehicle. Consider a simple rotating disc with a polar moment of inertia J turning at an angular velocity ω (rad/s) as shown in Figure 11.

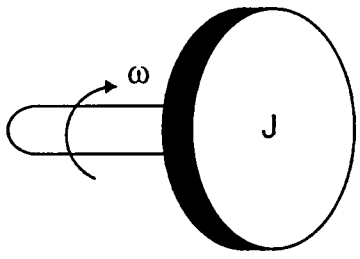


Figure 11: Rotating Disc Model

For this rotating disk, the amount of torque required to accelerate its rotation at a rate of α (rad/s^2) is

$$T = J\dot{\omega} = J\alpha \tag{3-8}$$

Because torque is related to force by the radius of the disk, the input force required at the periphery of the disk is thus $J/r\,\alpha$. Likewise, the same analogy can be made using an energy balance standpoint. While a moving component has

kinetic energy equal to $\frac{1}{2}mv^2$, a rotating (and thus moving) component has total energy equal to $\frac{1}{2}(mv^2 + Jw^2)$. Therefore, all of the terms associated with acceleration of rotating vehicle components must be considered during the computation of acceleration force.

Because the equivalent inertial mass of each vehicle may be different and exact values of polar moments of inertia may be difficult to ascertain, approximations may be used based upon previous studies. This value will change with gear ratio; however, a weight propagation factor of 15% will be used throughout this study [20]. In other words, the inertial mass will be assumed to be 15% higher than the physical mass (which is an excessively large value).

With all the terms of road-load forces defined, the total required drive power may be computed by multiplying the required force by the vehicle velocity. Assuming zero grade ($\theta=0$), this reduces to:

$$P_{DRIVE} = \frac{1}{2}\rho_{air} C_D A_f V^3 + mgVC_r + m_i aV \qquad (3\text{-}9)$$

This equation therefore completes the physical picture of the power required at the drive wheels of the vehicle. It is important to note that the vehicle power plant must produce slightly more power to overcome losses associated with the vehicle's transmission. Typically, the values of efficiency are high (in the 90+ percent range) [10]; for simplicity this efficiency will be assumed constant and the final equation for required drivetrain power is:

$$P_{DT} = \frac{1}{\eta}\left(\frac{1}{2}\rho_{air} C_D A_f V^3 + mgVC_r + m_i aV\right) \qquad (3\text{-}9)$$

where η is the mechanical efficiency of the drive system.

Driver Model

With the physical constraints defined, the next important element in the determination of the required drive power is the driver of the vehicle. In the end, it is the driver who determines the desired speed of the vehicle used in (3-9).

In general, we may consider the driver to be a proportional controller which controls throttle position based upon the difference between the speed the driver would like for the vehicle to travel (V_{des}) and the speed it is currently traveling (V_{auto}). For example, if the driver of a (conventional) car traveling 50 miles/hour would like to accelerate to 60 miles/hour, the driver will depress the accelerator pedal and thus increase the throttle opening. The increase in throttle opening or throttle position (TPS) causes the engine to produce more power which causes the vehicle to accelerate. As the vehicle speed increases, our driver slowly decreases the throttle position until a new equilibrium point at 60 miles/hour has been reached. True driver operation is more complex, however, this approach should suffice for the approximate calculations used here. This total system is depicted in the block diagram of Figure 12.

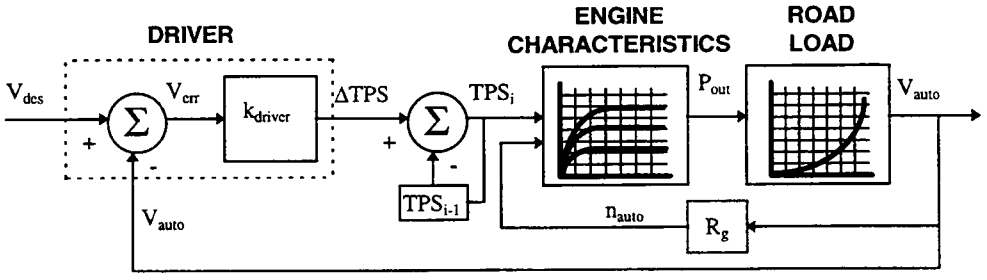


Figure 12: Driver and Conventional Vehicle Model

As shown, the driver proportionality constant, k_{driver}, is assumed to be constant.; it relates the error speed signal, V_{err}, to the change in throttle position, ΔTPS (for

simulation purposes, this change in TPS would be added to the throttle position at the previous time step, TPS_{i-1}, to determine the new throttle position, TPS_i). As will be discussed later, knowing both speed and throttle position on an engine will uniquely specify its power output which can then be used to drive the vehicle per equation (3-10). The transmission gear ratio, R_g, relates vehicle speed to engine speed.

Modeling System Components: the HPU

Fundamentals

The next term of (3-1) which must be understood is P_{HPU}. More specifically, understanding the relationship between the driver's input signal (i.e., throttle position) and the output power of the HPU will determine the power which must be provided by the EM.

First of all, it is important to understand that measurement of the output engine power (or torque), cannot be made without making unacceptable sacrifices in cost or reliability [23]. Therefore, an alternative, indirect method of measuring power must be found. As is expected, the induction of the air/fuel mixture into the cylinder of an i. c. engine controls the power output of that engine; therefore, intake manifold conditions must be understood in order to make predictions about the engine's power (or torque) output.

To that end, the manifold air pressure has been shown to uniquely identify an internal combustion engine's power output in steady state operation at a given engine speed and air-fuel ratio [23]. This has also been verified experimentally at the University of Tennessee as shown in Figure 13. The author performed tests on a Saturn SL which had been converted to natural gas; using a CNG-powered

engine was thought to provide information close to that of the HPU used in the UT-HEV.

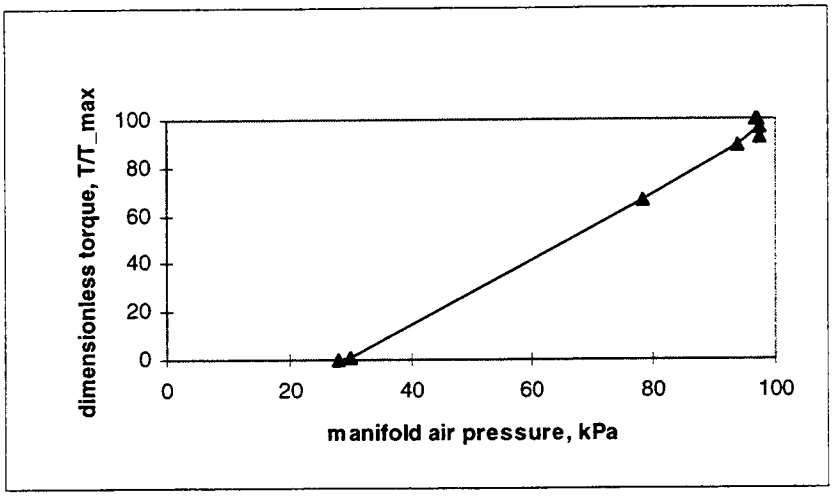


Figure 13: Manifold Air Pressure's Relation to Steady State Engine Torque Output

Therefore, once known, data relating the manifold air pressure (MAP) to output power can be curve-fit to provide a single equation. This method does ignore dynamic effects (the MAP-power relationship is valid at steady state). However, the time associated with the change in engine torque, in general, is much shorter than the time associated with the change in engine speed [23]; the later term's magnitude depends upon the inertia and thus the speed of the vehicle itself. Therefore, a linear (with respect to MAP), speed-dependent transfer function relating the change in manifold air pressure to output engine power can be found. Going back to the main requirement of this model, however, requires the derivation of a throttle position-based overall transfer function. It would seem logical to understand the connection between manifold air pressure and throttle position to reach this end.

Total Engine Model

The following discussion draws much of its material from a similar derivation by R.L. Morris, H. G. Hopkins, and R. H. Borcherts of Ford Motor Company found in their Society of Automotive Engineers (SAE) paper "An Identification Approach to Throttle-Torque Modeling" (SAE paper 810448, 1981) [24].

To understand this relationship, a simple model of the manifold, as shown in Figure 14, will provide the starting point for further derivation.

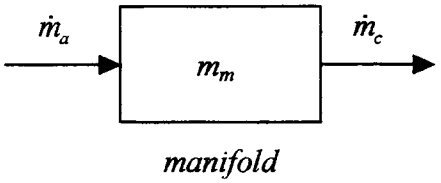


Figure 14: Manifold Model

The mass of air flowing in is represented by m_a (the mass of the fuel is neglected since the stoichiometric air/fuel ratio is on the order of 14), and the mass of charge (air) which is delivered to the cylinder is represented by m_c. Therefore, continuity through the manifold produces

$$\frac{dm_m}{dt} = \dot{m}_a - \dot{m}_c \tag{3-10}$$

where m_m is the mass of air in the manifold as shown in Figure 3. Following an ideal-gas assumption for the properties of the air inside the manifold (again, ignoring the effect of the fuel on this mixture) and substituting into (6) produces

$$\dot{m}_a - \dot{m}_c = \left(\frac{V_m}{R_m T_m}\right)\frac{dp_m}{dt} \tag{3-11}$$

where V, R, and T are the volume, universal gas constant, and temperature. The manifold air pressure, or MAP, is represented by p_m.

The next step is to look at the variables which affect the mass flow of air into and out of the manifold. The upstream mass air flow ($\dot{m}_a$) is related to both the throttle position θ and p_m. Likewise, the downstream mass air flow ($\dot{m}_c$) derives its value from both the manifold pressure and the speed of the engine, N (i.e., the rate at which the piston can accept the charge). In either case, standard linearization techniques produce the following equations (for small changes in manifold pressure, engine speed, and throttle angle):

$$\Delta\dot{m}_c = k_N(N, p_m)\Delta N + k_{pC}(N, p_m)\Delta p_m \tag{3-12}$$

$$\Delta\dot{m}_a = k_\theta(\theta, p_m)\Delta\theta - k_{pa}(\theta, p_m)\Delta p_m \tag{3-13}$$

Substituting (3-12) and (3-13) into (3-11) produces an expression for the change in manifold air pressure

$$\Delta p_m(s) = \frac{K_f}{\tau_f s + 1}\left[k_\theta(\theta, p_m)\Delta\theta(s) - k_N(N, p_m)\Delta N(s)\right] \tag{3-14}$$

where the manifold filling time constant, τ_f, and gain, K_f, are expressed as

$$\tau_f = \frac{V_m}{R_m T_m (k_{pc}(N, p_m) + k_{pa}(\theta, p_m))} \tag{3-15}$$

$$K_f = \frac{1}{k_{pc}(N, p_m) + k_{pa}(\theta, p_m)} \tag{3-16}$$

As is obvious, these terms each depend upon all three of the identified system variables: engine speed, throttle position, and manifold air pressure. Upon adding the effects of both engine inertia (also a first-order lag) and a "power stroke delay" (which accounts for the time between filling the cylinder and the production of engine torque; represented by pure delay e^{-st}), the final transfer function model for the engine is shown in Figure 15.

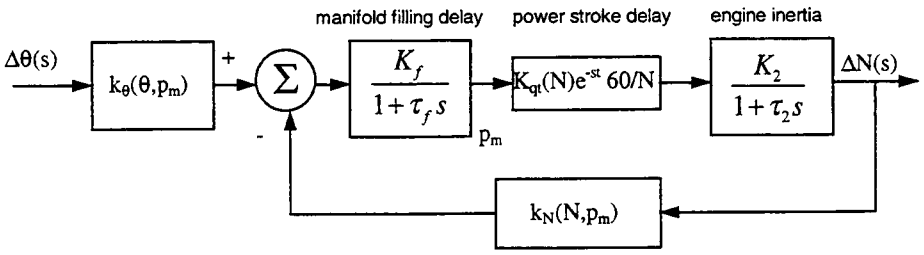


Figure 15: Engine Physical Model

The manifold filling delay transfer function produces the change in manifold air pressure; the next block includes not only the delay term, but also the transfer function $K_{qt}(s)$ which relates the manifold air pressure to the engine torque (the output of this block).

This method has been used with success and has been used to an accuracy of 3 N-m (in torque measurement)[23]. However, most of these tests had been carried

out at steady -state or with slow transients. For the purposes of this study, they will be assumed accurate.

Modeling System Components: The EM

Fundamentals

As discussed previously, the electric motor used in the UT-HEV is a permanent magnet brushless DC motor manufactured by Unique Mobility. Brushless DC motors do not run on DC current, as the name would imply, but instead they are multi-phase synchronous machines. The name comes from the fact that the operation of the motor can be made to resemble a DC shunt motor with a constant field current (i.e., when the motor is supplied with a voltage source at a frequency equal to the rotor speed) [25].

The following model of the brushless DC motor requires one major assumption to produce its linear form: the voltage source is assumed to have no phase shift. This leads to the following block diagram.

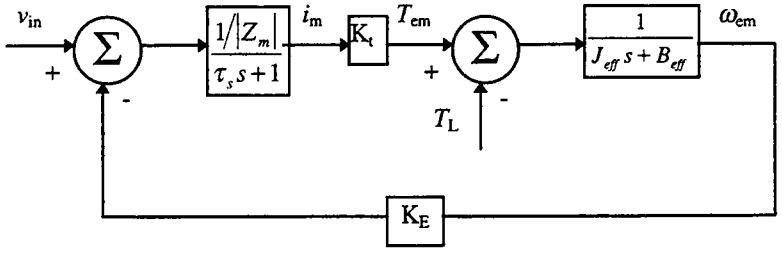


Figure 16: Brushless DC Motor Linear Model (adapted from Krause, Paul C. et al. *Electromechanical Motion Devices* McGraw-Hill, 1989)

Voltage input to the system is controlled externally and will be discussed later.. From the figure, the following expressions will be defined. First, v_{in} represents the

input voltage to the motor itself (this value will later be differentiated from the source voltage, v_s). The motor time constant can be expressed as

$$\tau_s = \frac{L_m}{R_m} \tag{3-17}$$

where R_m and L_m are the motor winding resistance and inductance respectively.

The overall motor impedence takes the form

$$|Z_m| = \sqrt{R_m^2 + (\omega_{em} P L_m)^2} \tag{3-18}$$

where P is the number of poles of the motor. Continuing through Figure 16, K_t is the motor torque constant that relates the motor current, i_m, to the output motor torque, T_{em}. Also, the load torque is represented by T_L, the effective inertia and damping of the motor and its connected load are defined as J_{eff} and B_{eff} respectively, and K_E is the motor back-EMF constant.

The motor used in the UT-HEV—the Unique Mobility SR-180P—may be characterized by the values shown in Table 3.1 for each of the above constants [26].

Table 3.1: Unique Mobility SR-180P Motor Parameters

Parameter	Units	Value
Motor EMF Constant, K_E	V/Krpm	27±10%
Motor Torque Constant, K_t	in-lb/Amp	2.2±10%
Winding DC Resistance, R_m	Ω	0.015
Winding Inductance, L_m	μH	25
Number of Poles, P	-	18
Rotor Inertia	in-lb-sec^2	0.200

Motor Speed Control

The control of any motor can be based on either controlling power output or speed; in this study, speed control is employed. Therefore, any motor model must also concern itself with not only the motor itself but also the motor controller. The motor controller (EMC) used in this study is the Unique Mobility CR20-300A controller. Its purpose is two-fold: first, the EMC has a power inverter which handles the conversion of the DC current from the ESS to three-phase, sinusoidal current used by the motor. Second, the EMC carries out speed control through the used of a pulse-width modulated (PWM) power amplifier.

As in a DC shunt motor, speed control can be achieved by varying the applied voltage; this is most often carried out through the use of pulse-width modulation [25]. Any PWM amplifier must employ at least two inputs and one output; the power supply or source voltage, v_s, and a control signal, L_{in} make up the inputs, the applied motor voltage, v_{in}, is the output. The control signal is associated with the switching of the voltage on and off to apply an average output voltage; the

details of the process will not be dealt with here. For more information on PWM, the reader is encouraged to read [27]. Therefore, a simplified, linear model of a bipolar PWM amplifier (which ignores effects of delay phase) is shown in Figure 17 [27].

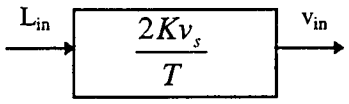


Figure 17: Simplified Model of PWM Amplifier

In Figure 17, K is a simple proportional gain; this relates input voltage to the time of duration of one phase in the PWM. The variable T represents the switching period of the amplifier. In addition to the other assumptions mentioned, this model also assumes constant source voltage; this may not always be the case since the characteristics of the ESS play a role in this value.

For any EMC, however, the value of the control signal L_{in} must be determined. The Unique CR20-300A uses both speed and current feedback as well as an external speed command to determine this control signal. The overall system is shown in Figure 18 coupled with the motor and amplifier models of Figures 16 and 17.

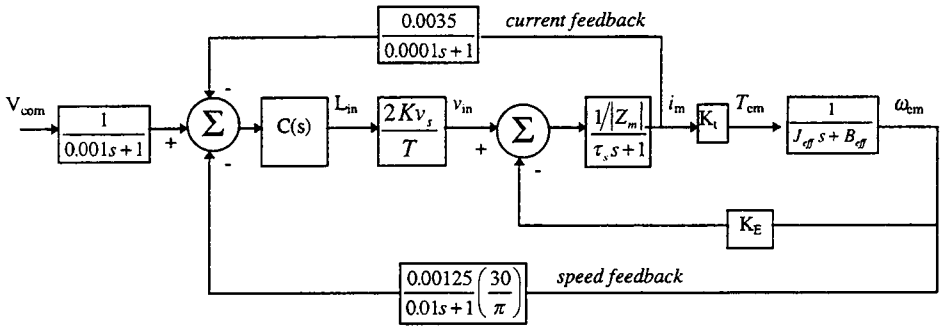


Figure 18: Complete Motor Model

The commanded speed input, V_{com}, serves as the primary control of the motor system. This signal is first filtered and then applied to the motor current and speed feedback signals. An error speed signal, V_{err}, is generated; it next passes through compensation and modulator circuits; their transfer functions are shown in equations 3-19a, b:

$$Compensation = C_1(s) = \frac{16.5(1+0.0033s)}{(1+0.109s)(1+3.3\times10^{-5}s)} \qquad (3\text{-}19a)$$

$$Modulator = C_2(s) = \frac{0.2}{(1+0.00013s)} \qquad (3\text{-}19b)$$

The transfer function C(s) in Figure 18 is the product of (3-19a) and (3-19b) (i.e. $C(s) = C_1(s)\, C_2(s)$). Therefore, the electric motor model has been fully defined.

Modeling System Component Control - The EM

Motor Power and Torque Control

To answer the three control questions associated with the EM - enable/disable, mode, and power - the system controller must interact with the EM and thus the EM model as shown in Figure 18. First of all, the motor may be assumed to be

enabled at all times during vehicle operation. Therefore, the next issues involve power and mode of operation.

For the motor used in this study, as previously mentioned, control of the motor power output (at least indirectly) is achieved through control of the commanded speed input (V_{com}) to the EMC. Additionally, the mode follows the same pattern. Should the input speed signal be less than the sum of the current and speed feedback signals to which it is compared, then L_{in} becomes negative. An additional regeneration level command further modulates the motor power in regeneration mode; setting this regeneration level to its highest setting provides maximum regeneration current while a regeneration level of zero produces no regeneration current [28].

To satisfy equation (3-1), the system controller must have the ability to control motor power (which, for a constant speed, amounts to motor torque control). Again, this equation applies only for the case of the operation in hybrid mode; operation as a pure electric (ZEV mode) does not require power control. However, since the EMC uses a speed control strategy to direct the EM, some means of manipulating the aforementioned system inputs must be devised to meet this end. The system which has been developed to meet this end involves a current feedback approach which continually adjusts the commanded speed signal based upon desired motor torque.

The torque-control block diagram of Figure 19 shows that three inputs are required to generate the correct commanded speed output signal. As previously mentioned, the motor torque output is directly proportional to the motor current; therefore, motor torque control amounts to motor current control. The values of motor current and torque will thus be used interchangeably.

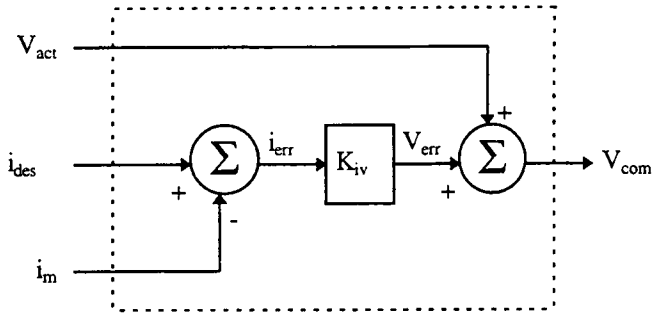


Figure 19: Motor Torque (Current) Control Block Diagram

A desired current (torque) signal, i_{des}, provides the system controller its only input into the block; both the motor current, i_m, and the current and speed feedback signals (summed to form V_{act}) come from the EMC. First of all, the desired motor current and the actual current are subtracted to form an error signal, i_{err}. Based upon data given by Unique Mobility, the motor current can be directly related to the difference in the commanded speed signal and the sum of the current and speed feedback terms (Unique manual); the constant of proportionality is the system gain which relates the current error signal to this speed-control error signal, V_{err}. Looking back to Figure 18, this relationship seems feasible—the error speed signal determines the value of L_{in} which in turn controls the motor current and torque output. Combining Figures 18 and 19 produces the final model of the EM including the torque-control scheme as shown in Figure 20.

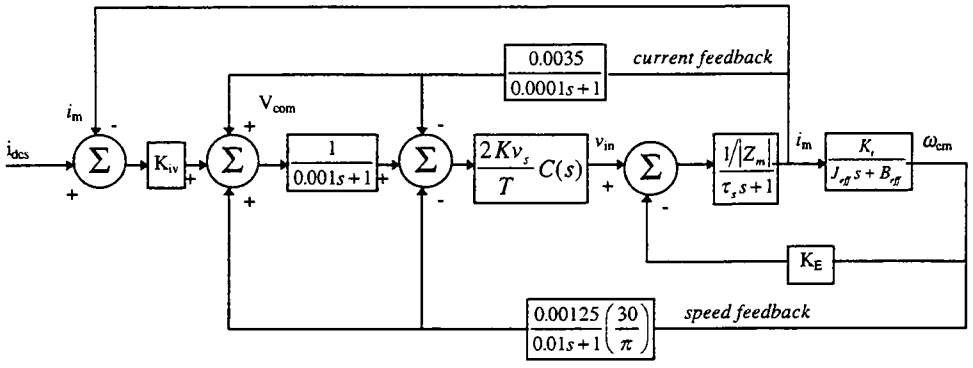


Figure 20: Complete Motor Model Including Torque/Current Control

Figure 20 shows that for the total control system, two parallel sets of current and feedback loops are required. This is due to the need to reconstruct the error speed signal; the control system exterior to the EMC (which includes the first two summing junctions) must "undo" some of the EMC's internal speed control functions in order to provide proper torque control.

Determination of Power Requirements and Vehicle System Integration

Motor Modes of Operation

Once the motor torque/power control has been established, some method of determining the amount of power needed under various operating conditions of the vehicle must be found. Motor operation falls into three important categories and analysis of motor operation in the context of the entire vehicle system is best handled by discussing each separately. Looking again to the total vehicle system, these modes follow from three possible states of motor power in equation (3-1). These relationships are shown schematically in Figure 21.

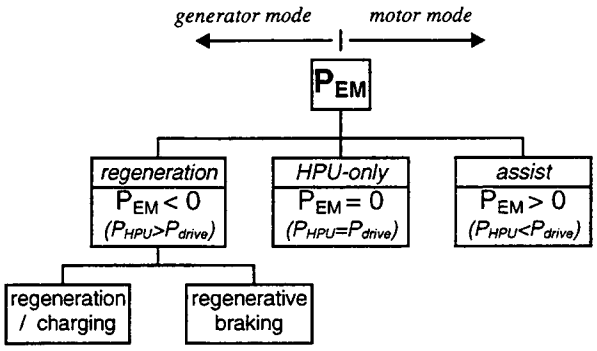


Figure 21: State Diagram of Motor Modes of Operation

Should HPU power be less than the requested drive power, (3-1) requires that $P_{EM} > 0$; this type of motor operation is called electric assist or simply assist. When the HPU power lies very near the requested drive power, the electric motor need not supply any power. Finally, should the available power from the HPU be greater than the requested drive power, a state of regeneration ($P_{EM} < 0$) is possible. This state could be reached under cruising conditions (perhaps during light deceleration) or during braking (where part of the regeneration energy comes from the vehicle inertia and not the HPU directly). These first and last types of motor operation will be discussed separately in regards to system control.

Determination of Power Requirements - Assist

The first step in defining this operation is to determine the state transition into the assist mode of motor operation from HPU-only operation. Obviously neither mode occurs at a point but instead over a wide range. HPU-only operation should end and electric assist should commence when the HPU reaches its maximum power output; beyond this point, the required motor power in (3-1) will surely become positive and assist will be required. Therefore, any indication

that the HPU has reached its maximum power is sufficient to determine this point of transition.

As previously discussed, manifold air pressure (MAP) demonstrates the most direct link to HPU output power (or torque); its linear correspondence to power makes it very attractive for this role. Throttle position, the chief system input from the driver, also can be related to the output engine power at steady state. However, a non-linear relationship to power as well as variation in power due to speed and MAP effects make throttle position less than ideal when accurate HPU power prediction is required. Therefore, in general, manifold air pressure would be the most likely choice to provide an indication of the transition point between HPU-only operation and electric assist.

The UT-HEV provides a unique opportunity, however, which makes throttle position an attractive choice for control of this transition point. Experimental testing has shown that the HPU produces maximum power output at a throttle position of approximately 25% for all engine speeds.; this data (unscaled due to dynamometer calibration problems) is shown graphically in Figure 22. This somewhat unexpected relationship owes its behavior to the oversized throttle-body mounted to the HPU; at one-quarter throttle, this large throttle body allows as much air to enter the engine as the original throttle body would when it was fully open. Consequently, the HPU can accept no more air (and thus produce no more power) after this low throttle position. Therefore, it would seem logical to use this throttle position as the threshold value required to determine the transition from HPU-only operation to electric assist.

Once the transition to electric assist has been made, the next step involves the determination of the required motor torque. Again, the TPS signal proves invaluable in this task: above 25% throttle, the TPS signal can be used by the

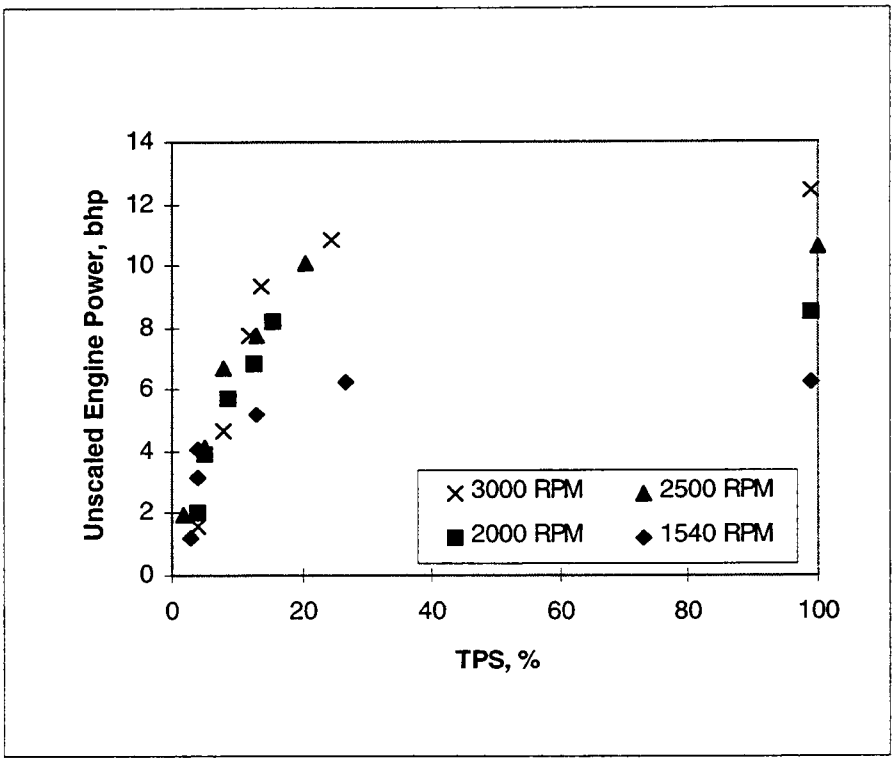


Figure 22: UT-HEV HPU Throttle Position / Power Relationship

driver to control motor torque much like the first 25% of the throttle control engine torque. In this way, the system controller has the ability to construct a composite power curve which gives the driver the feel of a normal vehicle by providing ever increasing powertrain torque output with increasing throttle position. This method of using thottle position to control motor torque has also been used by General Motors for a pure electric vehicle, the EV-1 [29]. An example of this composite torque curve is shown in Figure 23.

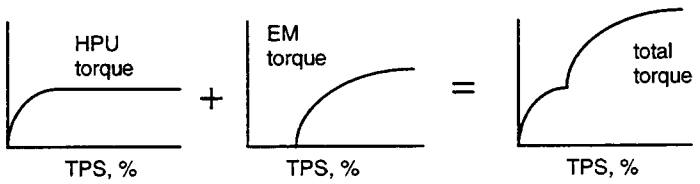


Figure 23: Construction of Composite Torque Curve for Assist

The next area of concern deals with the relationship between throttle position and desired motor torque output. The chief concern lies in the realm of driver feel and thus total system response (assuming that the maximum total power provides sufficient performance). Therefore, several functions relating desired current/torque to throttle position will be evaluated through means of simulation.

Three functions have been chosen for evaluation by simulation: step function, hyperbolic, and linear. In each case, the desired motor current is 0 A at 25% throttle and 200 A at 100% throttle. Figure 24 provides a graphical view of these functions.

The step function represents the extreme in possible assist strategies. Its inclusion in this study can be traced to previous efforts in the UT-HEV in which this strategy did, in fact, see use.

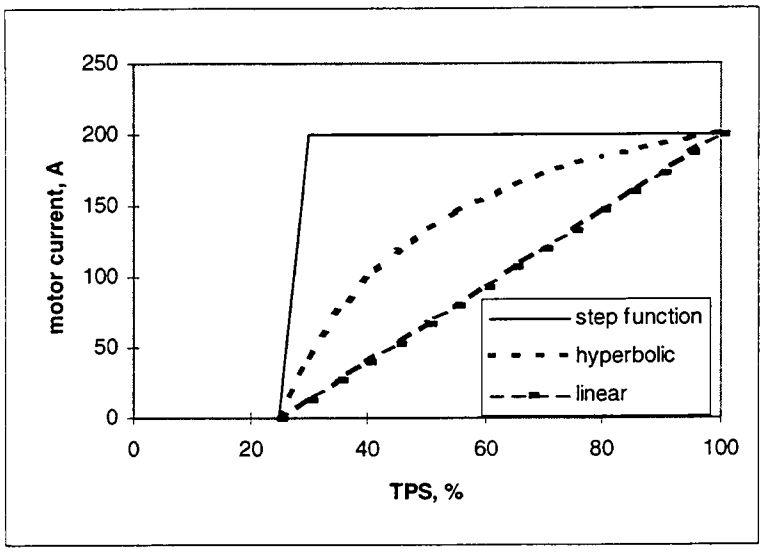
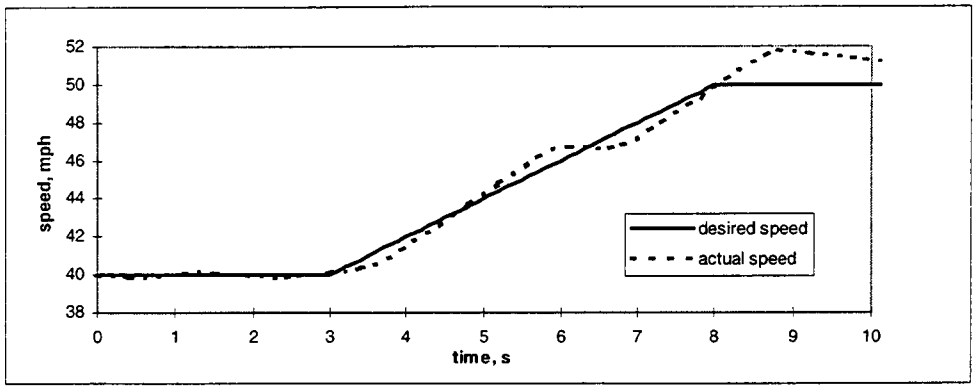


Figure 24: Assist Functions to be Evaluated

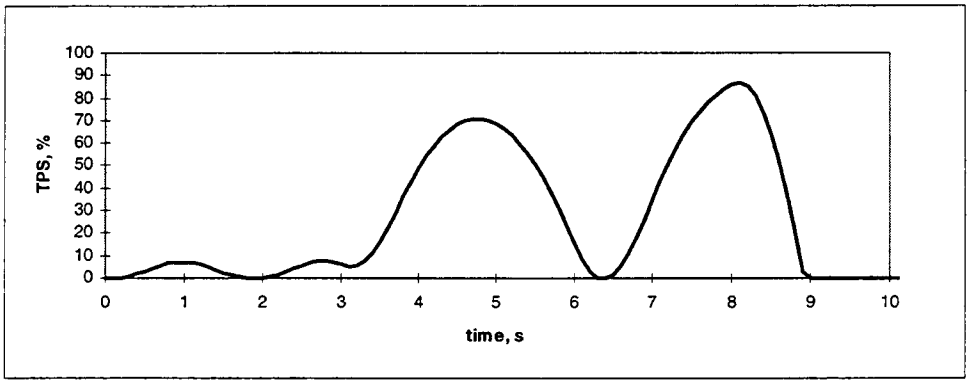
The linear function seemed the simplest choice for a real-life control system, while the hyperbolic assist curve seems a closer match to the HPU power curves. The hyperbolic might, therefore, have the greater chance of producing a smooth composite curve. This curve is defined by the equation $i = a + b / TPS$.

In order to evaluate these functions, a simple driving cycle has been chosen for the simulation; this cycle merely models highway acceleration from 40 to 50 mph. The same parameters in the simulation (which is discussed in appendix A) are used for each test.

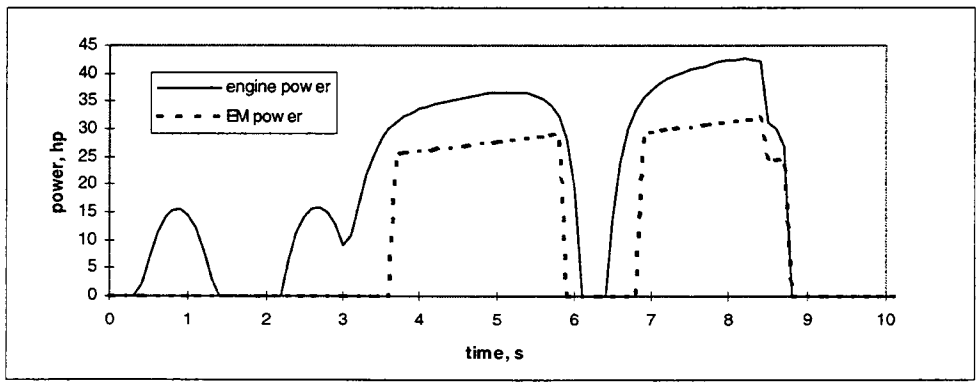
The most important measure of an assist strategy's success is it speed response; no driver wants to drive a car that cannot be controlled in a predictable or at least normal manner. Though this response owes much of its behavior to simulation parameters (especially simulated driver behavior), general conclusions can be drawn. Figure 25 first shows the response characteristics of the step function assist curve.



(a) vehicle speed response



(b) driver throttle command



(c) powertrain component power output

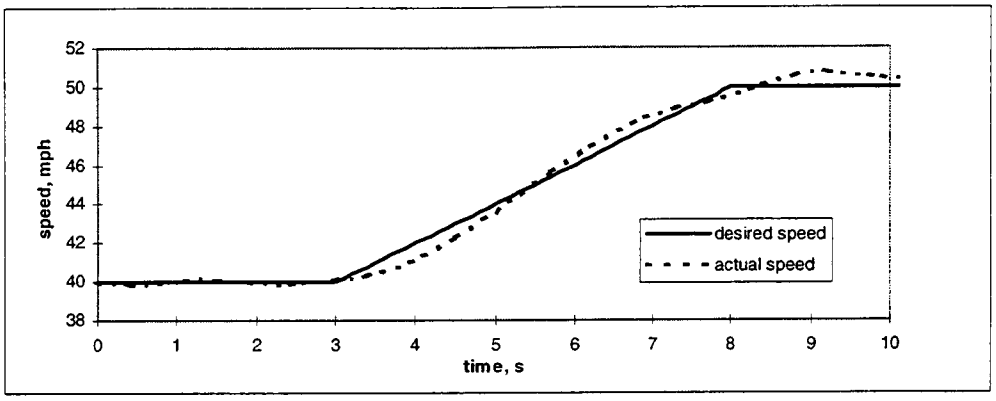
Figure 25: Simulated Vehicle Response with Step Function Assist Strategy

In Figure 25 (a), the vehicle speed response seems somewhat choppy then compared to the desired speed signal. This "choppiness" would be felt as jerking acceleration to the driver and would be undesirable. However, as shown by Figure 25 (b) and (c), this strategy does not require the driver to dictate full throttle operation; also, much of the power used in this cycle (when compared to other strategies) is supplied by the EM.

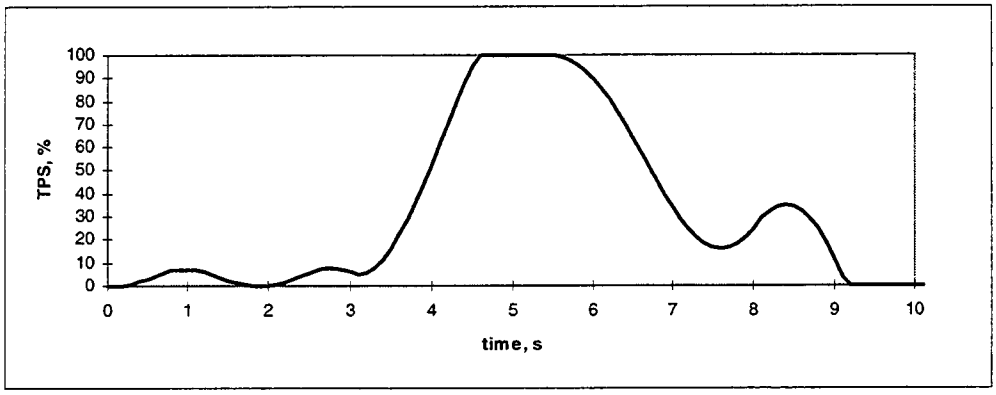
Figure 26 shows the same results when using the linear assist strategy. The smooth speed response curve of Figure 26 (a) demonstrates the ability of this strategy to provide a much more acceptable acceleration. However, full throttle is required and much more of the energy must be supplied by the HPU; this holds the consequence of reduced fuel economy during acceleration.

In Figure 27, the hyperbolic assist curve's response shows seemingly little improvement over the step-change function. The instantaneous rate of change of speed is less than that of the step function strategy, but the system overshoot is much the same. In comparison to the linear case, however, the EM provides a greater contribution to the total power output during acceleration thus leading to (in general) improved fuel economy over this cycle.

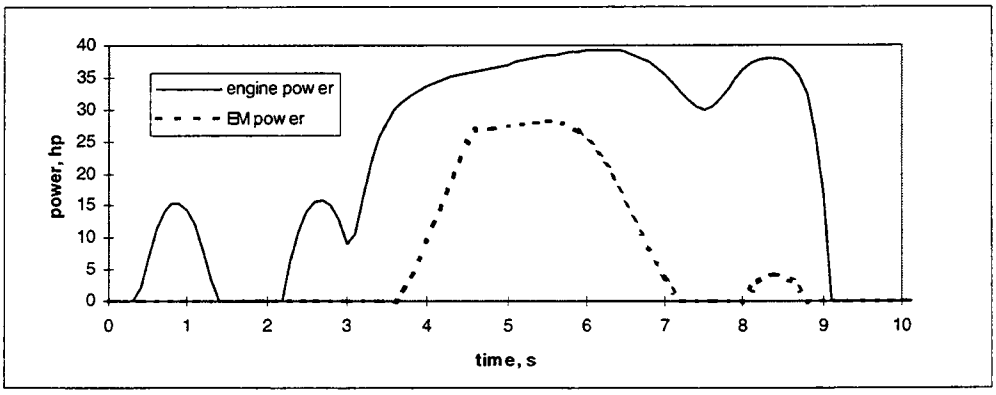
In conclusion, a trade-off seems apparent in control strategies: driver feel and fuel efficiency. Taking into account the fact that this cycle is only meant to serve as an sample case and not as the definitive example of HEV performance, the final decision for assist strategy will be determined through experimental means using multiple driving conditions; these may prove significantly different from the single case used for evaluation purposes here. Additionally, some of the variation in response may be due to the simplicity of the driver model.; definite conclusions should only be made based upon experimental results.



(a) vehicle speed response

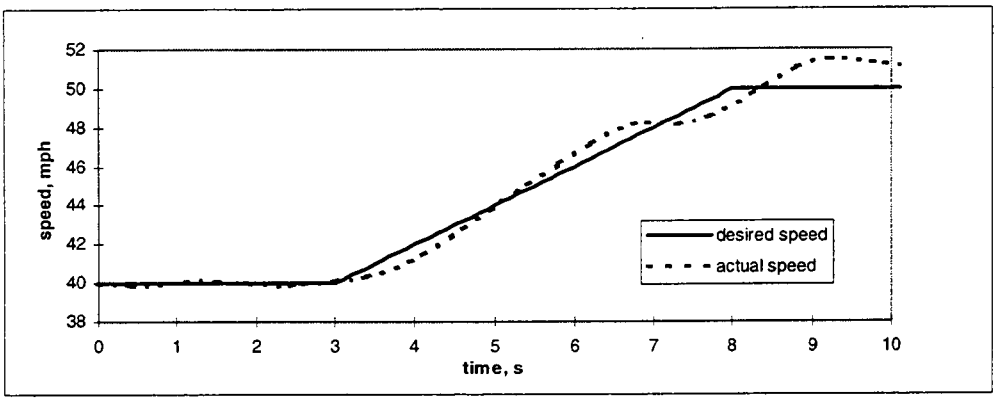


(b) driver throttle command

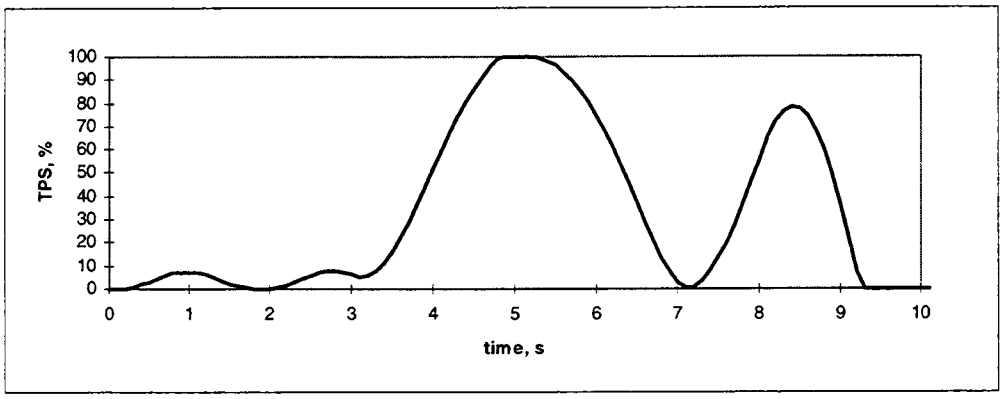


(c) powertrain component power output

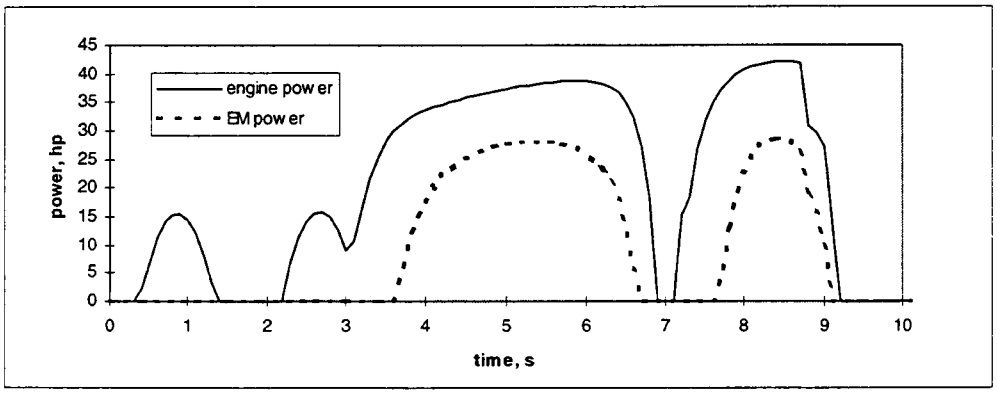
Figure 26: Simulated Vehicle Response with Linear Assist Strategy



(a) vehicle speed response



(b) driver throttle command



(c) powertrain component power output

Figure 27: Simulated Vehicle Response with Linear Assist Strategy

Determination of Power Requirements - Regeneration

Assist having been dealt with, the next area of concern lies in motor regeneration. Figure 21 shows two sub-categories of regeneration: regeneration/charging and regenerative braking. In the latter, it will be assumed sufficient to simply activate the EM in its generator mode at a constant rate of regeneration any time the brake pedal is depressed. Therefore, this section will focus on the more complicated issues of regeneration/charging (unless otherwise specified, simply referred to as regeneration) in which the engine drives the EM as a generator to charge the ESS. Equation (3-1) again directs the focus of the determination of the threshold determination: regeneration should occur when $P_{HPU} < P_{drive}$.

Using an argument similar to that used for assist, the threshold value can also be determined by the HPU's throttle position: at low TPS, the HPU supplies power at less than its maximum rate. Also, low TPS indicates that the driver does not need much power, or, in other words, the required drive power is low. Therefore, this condition is sufficient to determine the threshold value of regeneration.

Taking the argument one step further, the throttle position may also be used for the determination of regeneration level. Should the driver command zero throttle, for example, he wants the vehicle to slow down; maintaining a particular vehicle speed requires maintaining a non-zero throttle position. Therefore, as TPS decreases, the power requested from the driver decreases until zero throttle at which the "negative power" requested by the driver is maximum. Following this line of thinking, the regeneration level may be controlled by throttle position.

One physical limitation does, however, exist: at low engine (HPU) speeds, regeneration may have a detrimental effect: killing the engine. Therefore, TPS level should also be speed dependent. If TPS is below a certain threshold value

and engine speed is above a minimum, regeneration should occur, and the level of regeneration is determined by TPS.

Figure 28 shows the proposed method of regeneration control. Below a low threshold throttle position (here assumed to be 20%), the regeneration torque increases linearly up to a maximum at zero throttle.

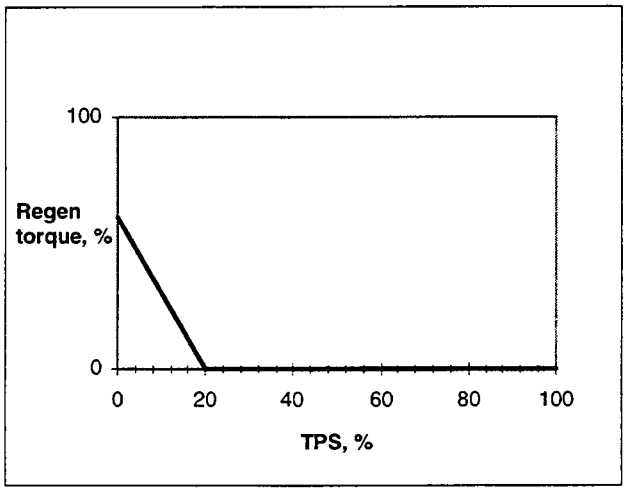


Figure 28: Regeneration Torque Control Strategy

Choosing 20% TPS as the threshold value provides a range in which the HPU operates alone to propel the vehicle at near its maximum power output. By increasing the level linearly (slowly) as TPS decreases, no abrupt decelerations should occur in transition from HPU-only operation to regeneration. Should the maximum level be too high, however, the driver will have little power at low TPS and will be required to always operate at the higher throttle positions where electric assist is present. This would be detrimental to vehicle operation since the balance in electric energy consumption and generation would be shifted to almost only consumption (assist). This balance is the cornerstone of charge-sustaining operation.

To ensure proper driveability and a sufficient balance of power (charge-sustaining operation), extensive experimentation will be required. However, simulation can shed some light on this subject.

Figure 29 shows the results of a simple deceleration from 30 to 20 mph in 3rd gear assuming a maximum regeneration current of 100 A and a threshold TPS of 20% for regeneration. As shown, this level of regeneration seems sufficient for gentle decelerations.

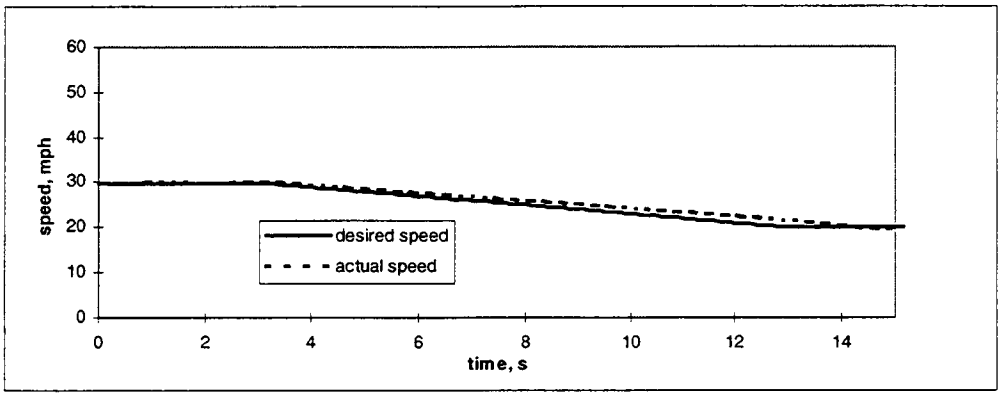
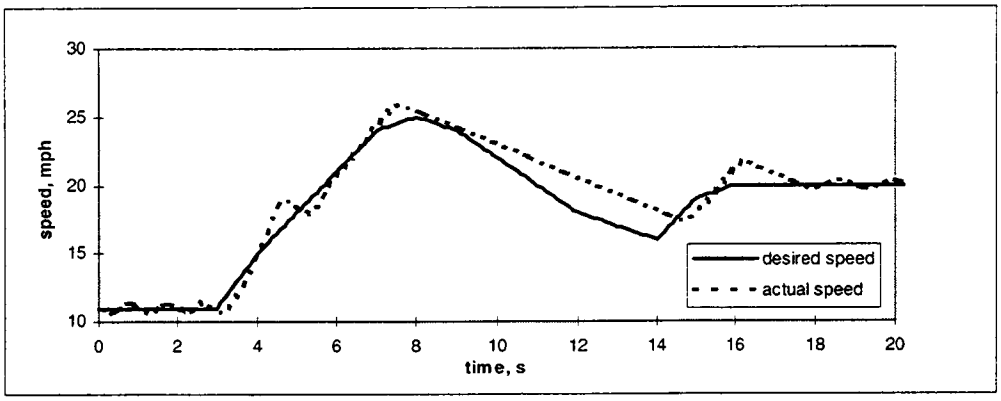
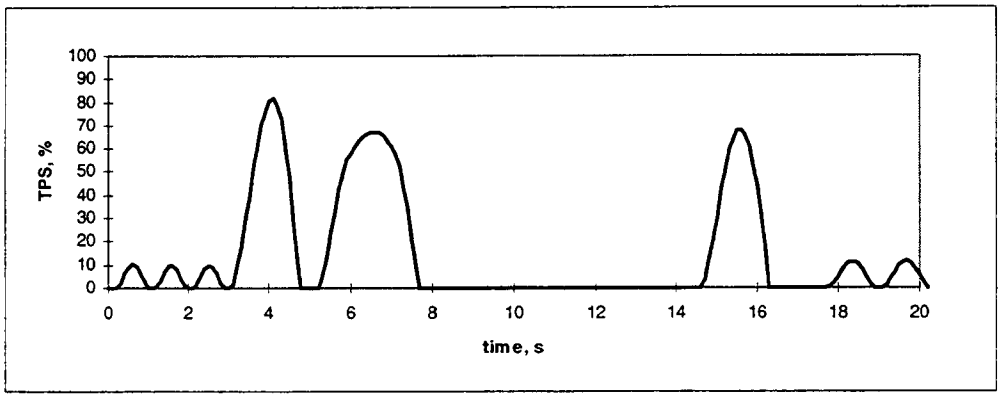


Figure 29: Simulated Vehicle Deceleration due to Regeneration (maximum regeneration current = 100 A, 20% TPS threshold for regeneration)

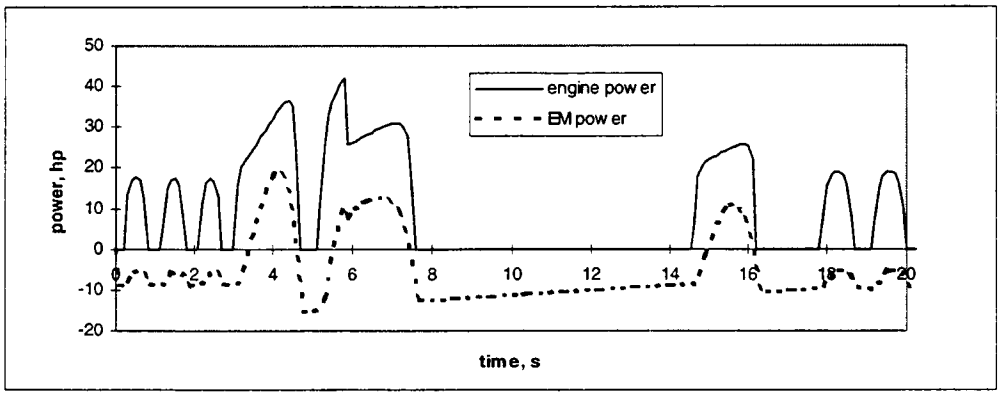
However, it is the interaction of acceleration and deceleration (where transitions from assist to regeneration occur) that is of most importance. Figure 30 thus shows the response of the total control system (using the linear assist model) to a random driving cycle involving acceleration and deceleration. It is important to again notice the effects of the simplistic driver model: the oscillations in both TPS and engine power between 0-3 s and 18-20 s can be directly attributed to the proportional controller's inability to maintain a steady-state throttle position.



(a) vehicle speed response



(b) driver throttle command



(c) powertrain component power output

Figure 30: Simulated Vehicle Response with Linear Regeneration Strategy and Linear Assist Strategy

The response of this strategy seems acceptable; many of the discrepancies between desired and actual speed come during gear changes. Also, as mentioned, the simulated driver has a large effect upon the results; therefore, final conclusions will be left to experimental determinations.

Complete System Model

As demonstrated in Figure 30, a complete and working simulation which includes both regeneration and assist has been developed. Therefore, the total system has been defined. Figure 31 presents the final and complete block diagram for the entire vehicle and control system. A few changes to the previous diagrams should be noted: first, the function $G_i(s)$ represents the assist control function which relates TPS to desired motor current (torque). The J and B terms shown represent vehicle loading through equivalent inertia and damping; these equivalent terms are velocity dependent and may be derived from (3-2) to (3-9). Each of the R terms represent gear ratios; in the case of the driver vehicle speed feedback loop, this term also includes the wheel radius to account for the conversion from angular velocity to linear (vehicle) velocity.

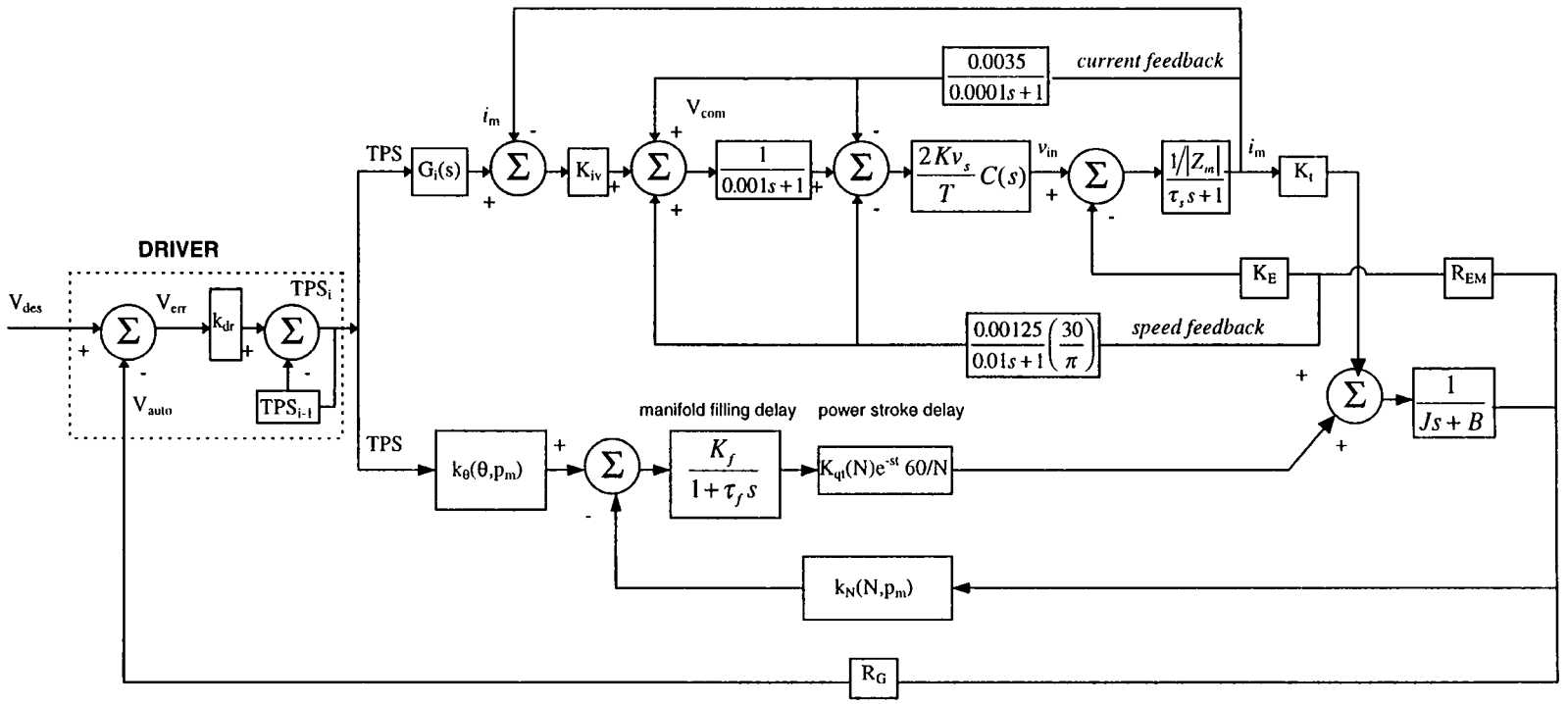


Figure 31: Complete System Block Diagram

Chapter 4

PHYSICAL APPROACH

Starting Point—The Original Control System

The UT-HEV's first-place performance in the 1995 Hybrid Electric Vehicle Championship indicated that its design had not been fully optimized. In fact, several key operating functions had yet to be implemented at the time of the competition. However, these shortcomings did not distract from the basic soundness of the vehicle; excellent emissions, a well-received design review, and an outstanding technical report helped the Neon across the finish line first. Coming into the second year of development, these faults had to be addressed.

Many deficiencies could be traced back to the original control system; several critical areas of functionality including regeneration/charging were not implemented at all. Controller flexibility and driver feedback, considered essential for the second year of competition, also cast doubts upon the ability of the existing system to meet project needs.

The control system employed for the 1995 HEV Challenge centered around a Motorola 68HC11EVB microcontroller. This controller handled all signal processing and decision making for the system; its control program was written in C and stored on an EPROM on the controller board. While this configuration provided a simple, low-cost solution to most of the control needs of the system, several problems became apparent. First of all, updating the system involved the complicated task of erasing and programming the EPROM every time a software change would occur; this process had to be accomplished away from the vehicle

at a site which had the proper equipment. Also, signal integrity and noise proved to be a problem for the motor speed control circuit; the EM exhibited random accelerations not related to driver commands. This made driving the vehicle as a pure electric almost impossible, especially at low speeds.

In the basic functions, opportunities also existed: first of all, while assist control was implemented, it proved very harsh; a large surge of power would occur whenever the assist would commence. And while regenerative braking was implemented, regeneration/charging was not. Finally, the system was not successful in monitoring battery state of charge or providing sufficient feedback to the driver.

Due to imperfections in both the implementation and the basic design of the first control system, the decision was made in the fall of 1995 to develop and implement an entirely new control system which could not only improve upon the original's user-friendliness and capabilities but also implement those important functions not seen in the original system.

Specification of the New Control System

Initial Concept

By its second year, the Neon's long-term role as a research vehicle had begun to emerge and take its place next to the immediate role of a competition vehicle. Specific goals and desires for a new control system quickly formed once the decision had been made to leave the old behind. Along with satisfactory implementation of those functions deemed essential to proper vehicle operation (electric assist, regenerative braking, regeneration/charging, operation as a pure electric vehicle, and monitoring battery state of charge), several new functions were added to the list: on-board data acquisition, driver feedback, and HVAC

control. As previously mentioned, the HVAC control function was eventually dropped.

Along with the execution of these enhanced control algorithms and new functions, several operational and organizational changes were also deemed necessary. First of all, ease of use and programming would require a much different approach to control system design. This goal was especially important when considering the needs of future research to update and optimize vehicle control settings. Also, reliability in wiring and signal transfer were also mandated. Finally, driver interaction through display of operating modes and conditions also became important.

With this broader scope for the new control system, it was decided that a microcomputer-based approach would best meet the needs of the vehicle. Also, because of the need of the system to manage several different tasks concurrently, a real-time operating system capable of simultaneously running separate but interrelated jobs was also deemed a system need.

Software and Hardware Requirements

The first area to be specified for the new control system was the operating system: QNX, the leading real-time operating system for the PC platform, was quickly chosen because of its price, industry status, and performance. Its compatibility with the standard PC platform also assured that the final hardware would be reasonably priced.

From the hardware standpoint, several focus areas drove the final decision. A 10 Hz execution speed was deemed the minimal requirement for system operation; meeting this performance requirement would ensure that the driver would not notice any significant lag in the response of the controller. Also, the hardware needed to be able to process all of the signals required to control the vehicle.

Therefore, a preliminary list of inputs and outputs to the system was developed; this list is shown in Table 4.1.

Table 4.1: Input/Output List for UT-HEV Control System

Function:	**General**			
Description	Name	I/O	D/A	Range
Ignition	IGSWI	I	D	12 V
Mode select - ZEV	ZEV	I	D	12 V
Dictator Assist	DICTA	I	D	12 V
Charge Sustaining Enable	DICTR	I	D	12 V
Regeneration Enable	REGEN_ON	I	D	12 V

Function:	**EM interface**			
Description	Name	I/O	D/A	Range
Commanded Speed	COM_V	O	A	0-10 V
Regeneration Level	REGEN	O	A	-10-0 V
Actual Speed	ACT_V	I	A	0-10 V
Enable Motor Controller	ENABLE	O	D	NA
Motor Temp. Alarm	EMTEMP	I	D	10 V
Motor Controller Temp. Alarm	MCTEMP	I	D	10 V
Motor Ready	EMREADY	I	D	5 V
Motor Controller Power	MCPOWER	I	D	5 V
Motor Current Limit	AMPLIMIT	I	D	5 V
Regeneration Indicator	REGEN_I	I	D	5 V
Direction Indicator	DIR_I	I	D	5 V
Fault Indicator	FAULT	I	D	5 V
Motor Current	I_MOT	I	A	0-10V

Function:	**Engine and driver interface**			
Description	Name	I/O	D/A	Range
Brake Pressure	BPRESS	I	A	0 - 5 V
Manifold Air Pressure	MAP	I	A	0 - 5 V
Throttle Position	TPS	I	A	0 - 5 V
First Gear Indicator	G1	I	D	12 V
Engine Speed	TACH	I	T	12 V

In Table 4.1, the first column gives the description of the control system signal; many of these will be discussed in more detail later. The signal's reference name follows in the next column. The column labeled "I/O" indicates whether the signal is an input to the system controller ("I") or an output of the system controller ("O"). Similarly, the "A/D" heading of the next column relates whether the signal is analog or digital; a signal may also be time-dependent and labeled with "T." Finally, the voltage levels in the last column provide further reference useful for signal conditioning or isolation purposes.

Based upon this information, the final control system had to meet the signal processing requirements as shown in Table 4.2.

Table 4.2: Analog, Digital, and Time-Dependent Signals Required for the System Controller

Signal Type	Input	Output
Digital	14	1
Analog	5	2
Time-Dependent	1	0

In addition to the specific input/output requirements, several general specifications regarding data storage, mounting, etc. were made; these are further outlined in Table 4.3.

Table 4.3: General Hardware Component Requirements

Component	Requirements
CPU:	Must provide adequate speed to run applications Must support QNX Must be rugged to withstand automotive application
Communications:	Provide communication with laptop computer in car for software development
Display:	Provide output to the operator Compatible device driver
Data Storage:	Contain operating system and control program Meet data acquisition requirements
Mounting:	Hold required number of components Provide power supply to controller Compact size Rugged design
Miscellaneous:	Input/Output terminal blocks Cables Opto-Isolation Modules Signal conditioning

These requirements formed the basis of the decision for the final control system.

System Controller Hardware

Choice and Specifications

The final choice for the control system was an industrial PC from Octagon Systems. Octagon was chosen as the primary vendor because of their use by other teams in student automotive competitions [30,31] as well as their approval and association with QNX Software Systems Ltd. Additionally, Octagon's hardware is designed to withstand a large temperature range (-40° to 85° C); its solid-state disks are also shock and vibration resistant [30]. A brief description of the components and system specifications is given in Table 4.4; a more complete description is given in Appendix B.

Table 4.4: System Hardware Choice and Specification

CPU	5025A (486SLC2 - 8MB)
	COM1, COM2 serial ports
	LPT1 parallel port
Digital I/O	5540 Multi-purpose card
	24 digital I/O lines
Analog I/O	5710 12-bit Analog I/O Card
	16 single-ended analog inputs
	2 analog outputs
	19 digital I/O lines
	2 counter-timers
Communications	*via 5025A serial port*
Display	5420 SVGA Video Card
Data Storage	5815 Disk Drive Card
	1.44 MB floppy drive
	510 MB hard drive
Mounting	5207RMH Card Cage
Power Supply	Computer Products NFC25-12T05-12

Comparing the specifications listed in Table 4.4 to those requirements laid out in Table 4.2, the UT-HEV control system did meet all data-processing requirements; the new system's 43 digital input/output lines, 16 analog inputs, 2 analog outputs, and 2 counter-timers along with additional serial and parallel ports enables it to meet almost any future need as well.

Along with the controller itself, several additional components made up the entire control system. First of all, many of the digital input signals had voltages higher than the 5 V logic used by the controller; refer to Table 4.1 for these values. To prevent damage to the controller, these signals were conditions through optical isolation modules. These modules not only provide correct interfacing of the differing voltage levels between sensors and the controller, but they also suppress noise in the system by eliminating a direct electrical connection between points.

Terminal blocks allow for direct connection of field wiring; this option is not possible with the pin terminals offered on the controller boards. The terminal blocks chosen employ screw terminals for attachment of field wiring. Each block is connected to the controller via multi-conductor ribbon cable.

A flat panel LCD display provides the needed feedback to the driver. The unit chosen is a Sharp LM32K07 which provides 320 X 240 resolution (one-quarter VGA) and is VGA compatible. This display was chosen over CRT's and other flat panel displays (both electro-luminescent and LCD) because of its small size (screen: 92mm X 122mm) and low price.

Additional Equipment

The system controller must interface with many devices throughout the vehicle; the characteristics of each dictate the interconnections between them.

First and foremost, the controller must interface with the electric motor controller. As previously mentioned, the particular EMC used in the UT-HEV is the Unique Mobility CR20-300A. This EMC provides control of the EM through four signals: enable, direction, commanded speed, and regeneration level. The remaining fifteen signals from the EMC provided information on the state of the motor as well as power and the system ground. As shown in Table 4.1, the commanded speed and regeneration level signals are both analog; motor controller enable as well as direction of motor rotation signals are both digital (the direction selector has been permanently wired to match the rotation of the HPU and therefore is not an handled by the system controller.). Apart from the actual speed and motor current analog signals, all other information from the motor is digital and used to ensure proper operation of the motor.

To gather information about the operation of the HPU, the controller must interface with the TEC2 engine controller. This unit, sourced from

Electromotive, controls fuel injection and spark timing for the HPU. To accomplish this purpose, it must monitor engine parameters such as manifold air pressure and throttle position as well as engine speed. Therefore, these signals are derived from this interface.

Information about the state of the ESS comes from the Cruising Equipment Kilowatt-Hour 2 Meter provided by competition organizers as a data acquisition system. This device provides information about battery voltage, current, and capacity used to the system controller via an RS-232 serial link.

Finally, the driver may directly supply information to the system controller via switches: HEV/ZEV mode control comes through a switch mounted on the dash of the vehicle; the first-gear switch is mounted just under the shift lever and is activated anytime the vehicle is in first gear.

All this information has been captured in the context diagram of Figure 32. In this diagram, the following new acronyms have been used: TX - transmission, DAS - data acquisition system (Kilowatt-Hour Meter), SWT - switch/junction box where the current shunt is located. The purpose of the context diagram is to show the signals coming in and out of the system controller (SYS CON) and the nature of each (digital, analog, or time-dependent).

System Controller Software

Design Concept and Basic Approach

The theoretical concepts in Chapter 3 address only the "what" of powertrain control; perhaps the greater challenge is the "how." As previously stated, the hardware at hand must carry out multiple functions correctly within time

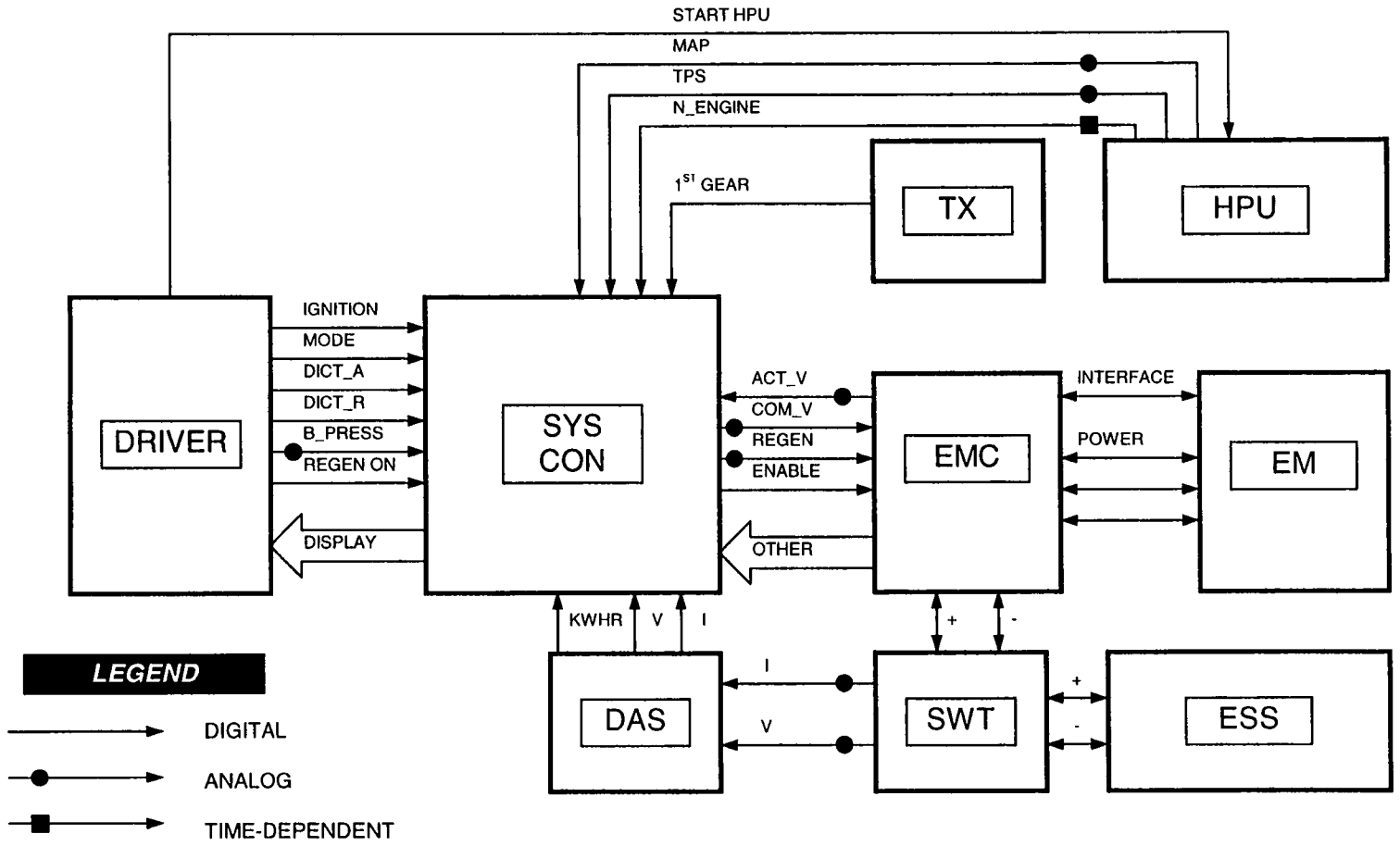


Figure 32: Context Diagram of Entire Control System

restrictions to ensure proper operation. This requirement defines the system controller as a so-called "real-time" system.

A more precise definition will shed more light on the subject:

> *Definition 4.1: "A **real-time system** is a system that must satisfy explicit (bounded) response-time constraints or risk severe consequences, including failure [32]."*

Obviously, poor response to driver commands in the UT-HEV could produce severe consequences that might put the life of the driver and others in jeopardy.

Research in the area of real-time systems has led to precise methodologies which have produced robust, deterministic systems. Several methods and techniques have been used in the development of this control system. The Real-Time Control System (RCS) architecture developed by A. J. Barbera and others at the National Institute of Standards provides a precise organizational structure and philosophy which has been shown to produce systems capable of managing complex real-time behavior [33]. Additionally, more traditional graphical tools such as data and control flow diagrams have also been used [32,34].

The basic goal of any software design project is the translation of the system needs into a requirement document and then decomposing this textual specification into layers of tasks; this basic approach has been taken in this project as is shown in Figure 33.

The RCS philosophy breaks down any control system into a few simple processes: sensory processing, value judgment, behavior generation, and a world model. The combination of these processes produces predictable behavior. In trying to graft the RCS philosophy onto this project, the author has determined

that a strict RCS approach would be more suited to a larger and more complex control system, however, some of the basic ideas still found their way into the UT-HEV control system.

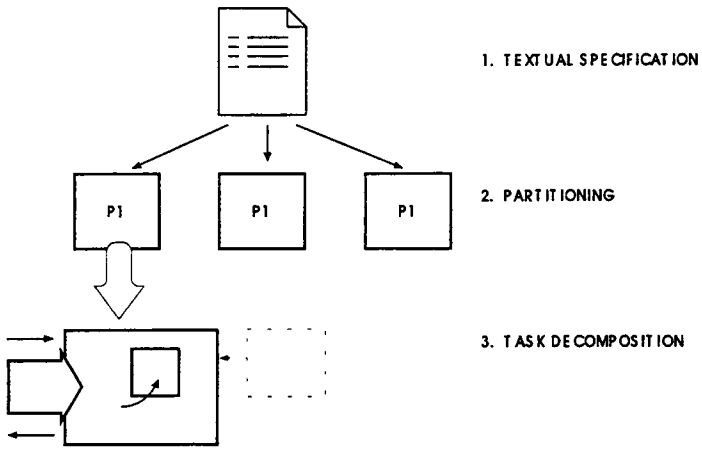


Figure 33: Software Design Philosophy

Employing the task decomposition, the control system may be broken down into two basic partitions: HEV operation and ZEV operation. Some commonality does exist between the two, however, it is simpler to separate the two due to the fundamental differences in the powertrain control.

Within the context of each operating mode, the structure may be decomposed into smaller tasks. In the HEV mode, powertrain control is the leading function at any point in time. However, several other functions are also important: monitoring the battery state of charge (SOC), providing feedback to the driver via the display, and acquiring and storing data for future use. These functions and their interactions are described in the data/control flow diagram of Figure 34.

Control and Data flow diagrams go one step beyond context diagrams in that they show, to some extent, the internal organization of data flow. Also, they may

demonstrate the flow of internal control signals. The solid lines represent data flow; dotted lines represent control flow. Single arrows indicate discrete data; double arrows indicate continuous data flow.

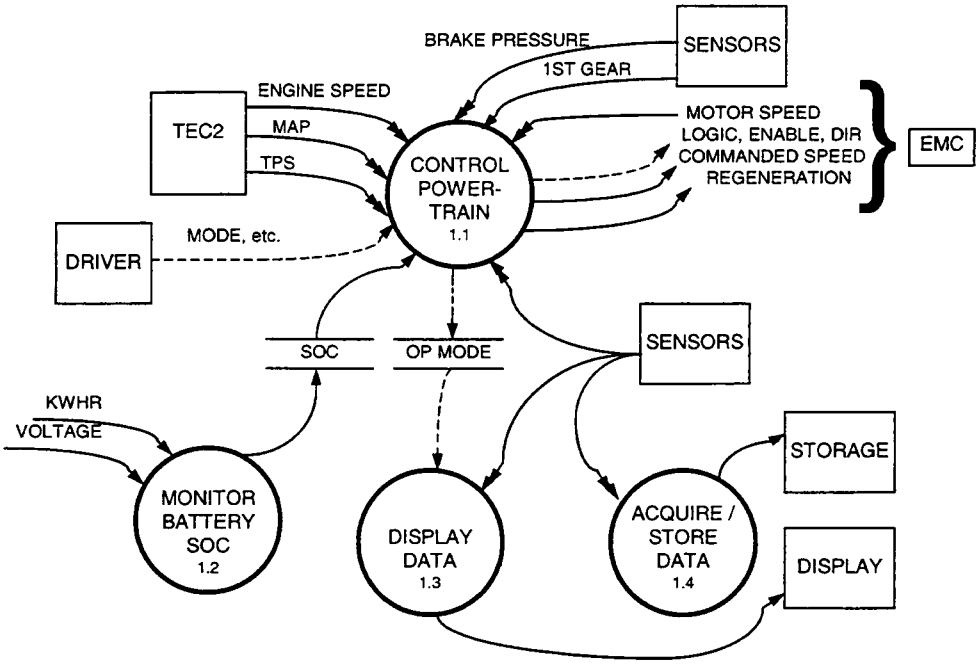


Figure 34: HEV Mode Main Control/Data Flow Diagram

Four basic functions have been defined for the overall control system: controlling the powertrain, monitoring the battery SOC, displaying data to the driver, and acquiring and storing data. It is obvious from the figure that powertrain control is the most complex function and thus dominates the others.

It is therefore powertrain control that drives most of the software design. Using the RCS concept of a rigid hierarchy with managers and subordinates, the concept can be developed. Obviously, the key system outputs are to the motor itself. Therefore, the lowest function in the hierarchy is the motor control

function; this function will accept commanded speed and regeneration level commands from its manager function and will place them in the proper form to be sent to the EMC. Another RCS tenet is the need for human intervention at each level; therefore, a regeneration disable function will also be incorporated in this function.

The next level of this behavior generation structure lies in the four possible operating states of the vehicle (in HEV mode): HPU-only, regenerative braking, assist, and regeneration/charging. In each state, the output to the motor would be different. For example, if the vehicle required electric assist, the assist management function would calculate the required level of commanded speed and send this to the motor control function along with a zero regeneration level command. However, if the vehicle required regenerative braking, the commanded motor speed would be set to zero and the regeneration level would be set to a specified value. In each case, the output is also driven by different input signals. Simply put, if the operating state is known, control of the motor is well-defined. The problem, then, lies in determination of this operating state.

The approach used to manage this state or mode choice uses a polling method that determines which of the states has exceeded the threshold value required for its operation. The approach is this: the mode operational manager (which may be considered to be a "third level" manager) reads in the necessary data to determine the thresholds for each of the possible operating states. It then polls the second level managers in sequence. In each case, this (second level) management function is given the threshold value and must determine whether or not its chief control variable has exceeded this threshold. In response, the function sends back its status to its manager. The first manager polled to have exceeded his threshold gains control of the motor control function (first level).

In this way, the RCS sensory processing and behavior generation functions at this hierarchical level are contained in these management functions. The second level (mode) managers not only direct the EM control function (behavior generation), but they also handle their own control signals (sensory processing). This structure is shown in the data/control flow diagram of Figure 35.

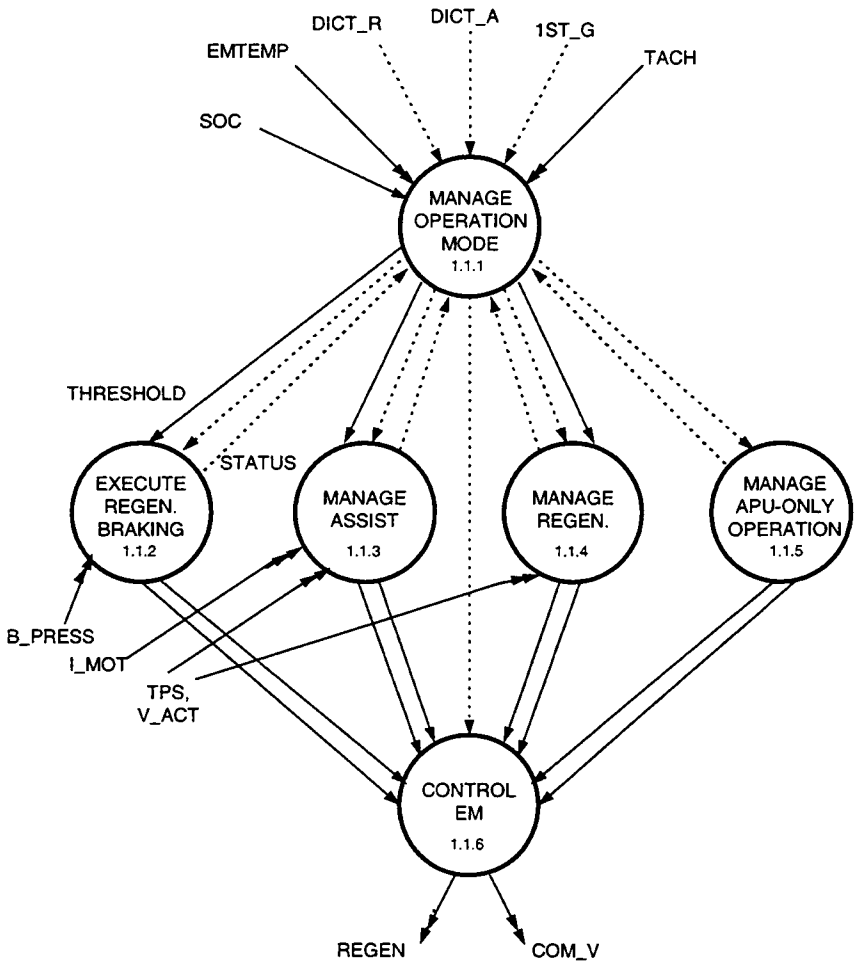


Figure 35: HEV Powertrain Control/Data Flow Diagram

Figure 35 shows the signal flows between functions. The operation mode manager (third level) sends the threshold value along with a status pointer. Each mode manager (second level) outputs both commanded speed and regeneration level to the EM control function. Finally, this function converts this numerical values to the voltage levels required by the EMC.

The polling structure also requires a hierarchy among mode managers. As previously stated, the first manager polled to exceed a threshold value gains control of the EM control function. This then implies that the polling order is important. In fact, the functions are polled in the order shown in Figure 35 (from right to left). This is also shown in Figure 36.

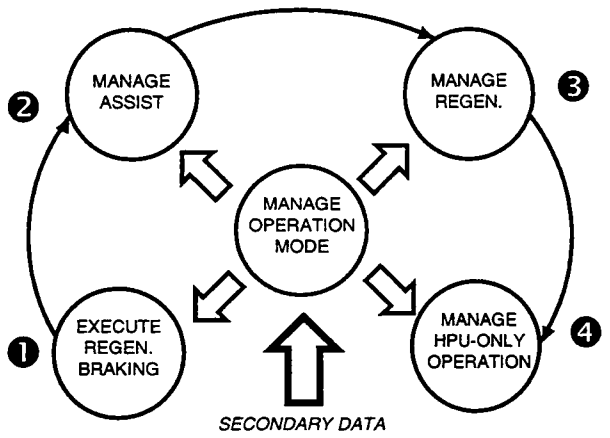


Figure 36: Powertrain Management Polling Order

This order is a reflection of the relative precedence of each of these functions. Braking comes first; should the driver be braking, none of the other functions should apply. Next, assist is considered more important than regeneration; in

fact, these modes are mutually exclusive. Finally, if none of the other mode thresholds have been exceeded, the controller reverts to HPU-only operation.

Threshold determination, which occurs at the third level, is the next area of importance. The throttle position threshold for regeneration, for example, could be changed depending upon driver requirements. Should the driver specify charge-depleting operation, the threshold could be set to zero throttle position which would completely disable regeneration/charging. In first gear, when assist is needed most, the assist TPS threshold is much lower than in normal operation. Also in first gear, the regeneration threshold is set to zero. Should a severe safety problem occur (such as an overheating condition in the motor), then the manager could set a limit on the assist threshold to reduce load on the motor.

Since these thresholds would be the most likely parameter that would need to be changed in the control system development, the different threshold values are stored in an external file. Whenever the main control program executes, it reads this external file. Therefore, changes to these values can be easily made without changing the control program itself.

To this point little has been said about ZEV operation. The main reason is that ZEV powertrain management is much simpler than HEV operation. In the ZEV mode, only two states exist: motor speed control and braking. Since these two states are mutually exclusive (brake pressure determines the transition between states), control is much simpler. In speed control, the controller samples the throttle position signal, filters it, and then sends the value to the EM control function (which is identical to the one used in the HEV mode). In braking, the controller transmits the commanded regeneration level signal to the same function. It is interesting to note that the current "brake pressure" is merely digital (the same signal that activates the brake lights); however, the software still

treats the signal as if it were analog. This enables future upgrades in this area to be made easily.

Final Software Configuration

Translation of the initial concept to the finished controller code shown in Appendix C did not occur simply or without some modifications. The first and largest change came in the operating system. Due to time limitations and difficult implementation questions, QNX was dropped in favor of MS-DOS 6.22. Despite the asynchronous nature of DOS, the implementation of the final control program seemed to suffer little. The control code, which was written in Borland Turbo C++, was still able to exceed the execution speed requirement. During HPU-only operation, the system executes at a rate of 46 Hz; during assist (which requires the maximum amount of analog to digital conversions), the execution speed is still within the 10 Hz guideline.

The change from a real-time system to an asynchronous operating system makes the direct implementation of the data/control flow diagram of Figure 34 impossible. However, to simulate the simultaneous operation which would occur in a real-time system, these functions are scheduled to execute at a fixed time interval (every 1 second). This places these lower-priority functions out of the queue for CPU time during 90+% of the computational cycle.

Chapter 5

TESTING APPROACH AND RESULTS

Testing Objectives

Once the hardware was installed and the software proved workable, the focus of the study shifted towards testing. The testing of the vehicle was conducted to meet two objectives. First of all, experimental data could provide the means necessary to properly tune the control system parameters. Some attempt to accomplish this task has been made through simulation. However, it was felt that only empirical evidence could verify these predictions. Secondly, the testing could evaluate the overall vehicle performance. Since the first research objective involves producing a vehicle whose performance is similar to that of a conventional automobile, this testing also serves as a means to do just that.

Testing Methodology

Informal Testing

Because the UT-HEV began as a working vehicle and has remained such during most of the time of this study, the testing approach involved a long period of informal testing which led to a control system which seemed worthy of the formal testing that was to follow. This informal testing will be dealt with in four time periods: initial vehicle evaluation (which occurred in the fall of 1995), American Tour de Sol (May, 1996), Chrysler Press Event (October 1996), and final vehicle evaluation (winter 1997). Each of these milestone events brought forth more information about the vehicle and the needs of the finished control system; the ability to perform this informal testing not only reduced the amount of time

required to perform the final, formal testing, but it also helped to produce a better finished product.

Formal Testing

The formal testing of the vehicle was designed to fulfill two purposes. The first involved the final determination of control system parameters. While the informal vehicle testing provided direction in this area, formal testing was still required to make the final decision. This would ensure an objective choice of operating parameters. Second, the formal testing would establish the performance of the vehicle to evaluate the project's success in meeting the first research objective as discussed previously. To accomplish this second goal, three testing areas were identified.

For the vehicle to operate like a normal automobile, the finished control strategy must not respond to driver inputs in an unexpected or unusual manner. Therefore, the first area of formal testing involved the evaluation of vehicle driveability. This was to be accomplished through subjective driver evaluation as well as through monitoring the speed response of the vehicle over certain driving tests. It is assumed that sudden changes in speed over these cycles indicates a poorly tuned control strategy; this will be discussed in greater detail later.

While driveability is certainly important, the whole point of a hybrid powertrain is to improve the overall energy efficiency of the vehicle. Therefore, the second area of testing centered around the investigation of how different control strategies affect the amount of energy used by the vehicle over a specified driving cycle.

Finally, the third area of evaluation is the vehicle's acceleration performance. However, time restrictions and the mechanical integrity of the vehicle's

powertrain have prevented this area of testing from being carried out at this point in time.

Testing Procedures

Informal Testing

Little needs to be said about the procedures used during the informal testing because of their largely subjective nature. However, they do provide a reference point for discussion of results.

Initial Vehicle Evaluation (fall 1995): Following the 1995 HEV Challenge, the vehicle underwent subjective evaluations through both road tests and through the use of the chassis dynamometer.

American Tour de Sol (May 1996): The competition for the second year of the Neon did involve formal testing at an EPA-certified test facility in Brooklyn, New York. This involved standard emissions testing using a chassis dynamometer. Because of failures associated with the EMC, none of these tests were entirely successful. Therefore, only a subjective evaluation of these tests will be discussed.

Chrysler Press Event (October 1996): At the invitation of the Chrysler Corporation, the vehicle traveled to the company's Technology Center in Auburn Hills Michigan in the fall of 1996 where it and several other student vehicles were subjected to review by the automotive press. This involved driving the vehicle around a 1.8 mile evaluation road at speeds of up to 60 mph. This marked the first occasion that vehicle had been driven on the road by people outside of the UT-HEV team. This peer evaluation provided excellent feedback about the vehicle's ability to meet the first objective of producing a car with normal feel and performance.

Final Vehicle Evaluation (winter 1997): Following the Chrysler event, the vehicle underwent many changes in preparation for its final testing in March. This again involved dynamometer testing of the vehicle.

Formal Test Procedure - Vehicle Driveability Tests

Each of the two formal test areas will be dealt with on an individual basis. First of all, vehicle driveability studies were carried out using the University's chassis dynamometer (this and other equipment is presented in detail in Appendix D). The dynamometer contains a set of rolls in which the drive wheels of the vehicle rest. As the drive wheels begin the rotate, they must also rotate the dynamometer rolls. These rolls are coupled through a belt to eddy current brake. The resisting torque of the dynamometer rolls may be modulated by changing the current to the eddy current coils; through this modulation the dynamometer can simulate the vehicle road load as discussed in chapter 3. In order to properly simulate this behavior, the user must specify two dynamometer settings: vehicle weight and aerodynamic properties.

Two limitations occur in using the UT chassis dynamometer. First of all, calibration of these dynamometer settings is difficult at best; in fact, no equipment is available to perform a direct calibration. Secondly, the dynamometer can modulate the resistance in the rolls to correctly simulate acceleration, but it cannot correctly model the deceleration of the vehicle. This limitation comes from the fact that the eddy current brakes would have to drive the rolls to simulate this inertia. Since they are incapable of driving the rolls, the sole loading factor on deceleration comes from the inertia of the rolls themselves. Unfortunately, the rolls provide only about half of the needed inertia to properly simulate vehicle deceleration. This becomes particularly important when considering the effect of regenerative braking where the inertial energy of the vehicle can be used to charge the ESS.

The main purpose of driveability testing is to compare different control strategies. Therefore, if consistent dynamometer settings are used through all the tests, then the validity of the results should still stand. Using this methodology and assuming that the dynamometer is calibrated correctly, the final dynamometer settings reflected the actual vehicle mass with the driver (3200 lbm) and aerodynamic properties ($C_dA_f = 6.6$ ft^2).

During these tests, the most important variable to monitor was the vehicle speed. This was accomplished by monitoring the speed signal from the dynamometer. This analog signal was sampled through an analog to digital converter at a rate of 2 Hz. For reference, this signal was found to match the vehicle speedometer reading. Other variables measured during these tests included motor current, the EMC's V_{act} and V_{com} signals, and the throttle position. The first three come from the EMC itself: motor current is measured via Hall-effect sensors in the motor while the other signals are generated by the logic board on the EMC. The throttle position comes from the throttle position sensor via the TEC2. In each case, these analog signals are sampled through the system controller's 12-bit analog to digital converter at a rate of 10 Hz. To satisfy the Nyquist condition, it is assumed that the frequencies of each of these signals is less than 5 Hz. The system controller transferred these values via an RS-232 serial link to a laptop computer where the final data storage took place.

To begin testing, the driver specified a particular driving cycle on the vehicle speed monitoring system. The program would display the driving cycle (desired speed) and the actual vehicle speed. Because of the human control of the actual speed, there is some variability in results. Therefore, several tests would be run to ensure the repeatability of the results.

For this testing, the following driving cycles were deemed important: highway acceleration (40-50 mph in 4th gear), deceleration (30-20 mph), first gear acceleration, and braking. For highway acceleration, both linear and hyperbolic assist control functions were analyzed. Deceleration involved only a linear regeneration strategy; however, the maximum amount of regeneration was modulated. A single point test was conducted in the remaining two tests to demonstrate their effectiveness.

Formal Test Procedure - Energy Efficiency Tests

Energy efficiency testing was also carried out using the chassis dynamometer; therefore, the same restriction applies: only comparative conclusions can be made. For these tests, the Federal Urban Driving Cycle (FUDS) was chosen because of its wide use in automotive research. This approximately 23 minute cycle is shown in Figure 37.

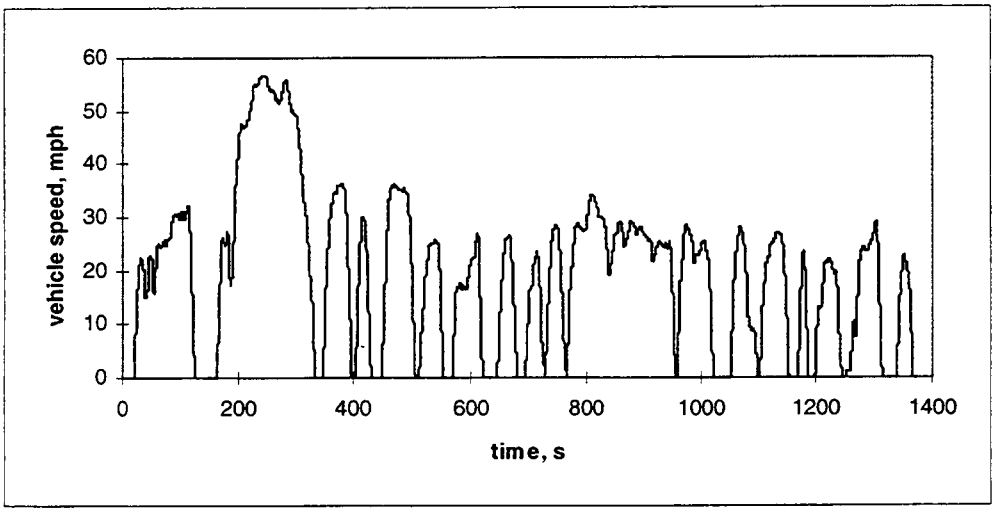


Figure 37: Federal Urban Driving Cycle

The same speed monitoring system was employed to assist the driver in following the driving cycle. The vehicle speed data was not recorded for this test.

The data collected in the test included total electrical energy consumed during the test, no-load battery voltage, and fuel temperature and pressure. All these values were collected from the on-board data acquisition system (DAS) provided by the competition organizers. Net electric energy was measured in kW-hr; the DAS came to this value through integration of the power used (voltage * current) over time. The pressure and temperature were measured using sensors mounted at the tank; these are described in Appendix D.

This test involved the evaluation of three possible control strategies: fully charge-sustaining (with regeneration/charging), charge-depleting (without regeneration/charging but with regenerative braking), and ultra charge-depleting (no regeneration whatsoever). In each case, the vehicle was driven over the driving cycle and the aforementioned data were collected. Tests were conducted more than once to obtain an indication of the repeatability of the results. The information collected could then determine how much each EM operating mode (assist, regeneration/charging, and regenerative braking) contributes to the total electrical energy consumption. Equivalent fuel economy was also calculated from beginning and ending states of the fuel; sample calculations are also contained in Appendix D.

Discussion of Test Results

Informal Testing Results

The initial vehicle evaluation in fall of 1995 provided several key points of information about the vehicle. First of all, the vehicle saw extensive road testing during this time. Most of this testing was conducted without electric assist due to control system failures. Subjective evaluation demonstrated that the vehicle could not operate in a "normal" manner by using solely the HPU to propel itself. First gear launches proved quite difficult (especially on hills) and highway acceleration could only be deemed abysmal.

By the time of the American Tour de Sol in May, 1996, the new control system had been installed. Despite failures in the EMC and mechanical powertrain coupling, the vehicle was able to perform through at least part of the testing. During this testing, one thing was apparent: the transition from HPU-only operation to assist was much too harsh. The vehicle experienced large surges of acceleration which were disconcerting to the driver. Future testing would have to address this problem.

The October, 1996 press event at the Chrysler Technology Center allowed the automotive press to provide their own feedback about the car. On the road course the vehicle's acceleration now seemed comparable to a conventional automobile. However, two of the drivers did note specific problems with the assist transition. First of all, the instantaneous surge of power which occurred at the assist transition was somewhat disconcerting and difficult to predict. Also, the vehicle experienced a limit cycle phenomenon at highway speeds. The driver could reach a point at which small modulation of the throttle could cause the assist to engage and disengage at a rapid rate. Finally, deceleration proved one other control system deficiency: every driver allowed the engine to die while pulling off of the course because they did not depress the clutch pedal in time to halt regeneration.

The final evaluation which led up to formal testing proved fruitful. Using the current feedback system discussed earlier, the assist transition roughness appeared to be under control. Also, the first gear assist strategy received an overhaul. Finally, a speed limitation was imposed on regeneration to prevent killing the HPU during driving.

Formal Testing Results - Vehicle Driveability Tests

For the highway acceleration testing, three control strategies were evaluated: assist 1, assist 2, and assist 3. Each of these strategies were defined by two parameters. First, the value of the current feedback system gain (K_{iv}) and the type of curve (linear or hyperbolic). Each strategy's characteristics are shown in Table 5.1.

Table 5.1: Assist Strategy Characteristics

Parameter	Assist 1	Assist 2	Assist 3
Current Feedback Gain, K_{iv}, %/A	0.05	0.15	0.15
Curve Type	linear	linear	hyperbolic

First of all, the results for the vehicle without assist on the highway test are shown in Figure 38. This data demonstrates the inadequacy of the HPU to propel to vehicle during acceleration. In fact, the actual acceleration of the vehicle is less than half that required by the driving cycle (< 1 mph/s vs. 2 mph/s).

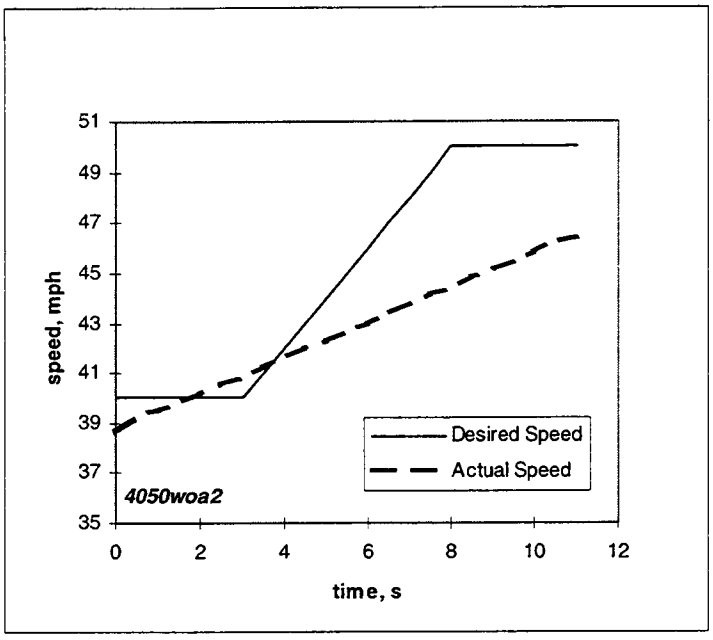


Figure 38: Acceleration Test Results Without Assist

The first attempt to produce a workable assist strategy (Assist 1) produced the speed results shown in Figure 39. While the actual results show that this strategy prevented the vehicle from meeting the speed requirements only marginally, the subjective driver evaluation was far worse. Monitoring motor current showed that the power from the motor was much lower than expected. Therefore, the current feedback gain was increased. Through rough testing and estimation, the new value of K_{iv} was determined to be three times that of the original. At four times this value, the assist transition was too rough.

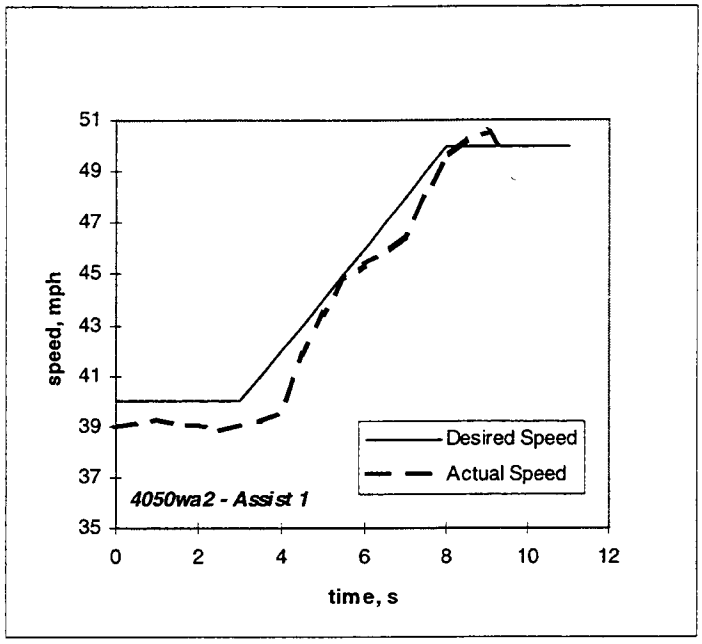
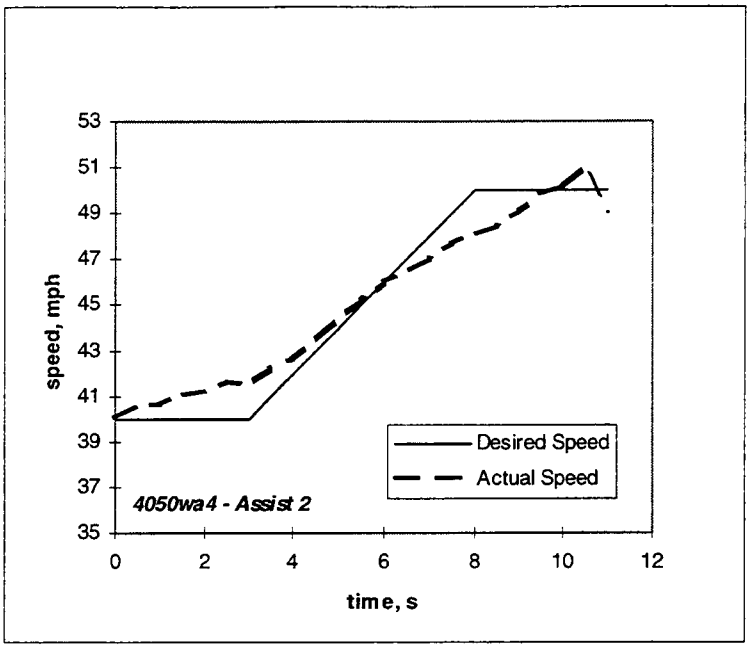


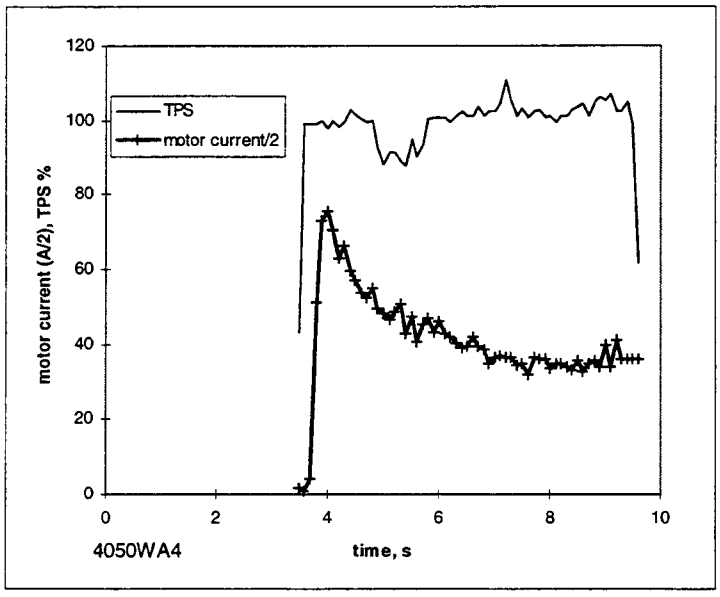
Figure 39: Acceleration Test #2 Results Using Assist 1 Strategy

Adjustment of the K_{iv} value led to Assist 2. On the same driving cycle, it subjectively showed a greater ability to provide the required acceleration than did Assist 1. Figures 40 and 41 show the results of tests 4 and 5 of the Assist 2 strategy. As demonstrated, acceleration, while smooth, still did not meet the required value. In fact, both tests show a 10-20% lag in this area. Also, the motor current in each case peaks initially at a value of approximately 160 A but decreases asymptotically towards approximately one-half that peak value.

The hyperbolic curve of the Assist 3 strategy was then employed to remedy the acceleration deficiencies seen using the Assist 2 strategy. Figures 42 and 43 demonstrate its success. By employing Assist 2, peak accelerations reached 3 mph/s (vs. 2 mph/s requirement). This owes much to the hyperbolic curve's ability to keep the motor current high; in both tests the average current remained

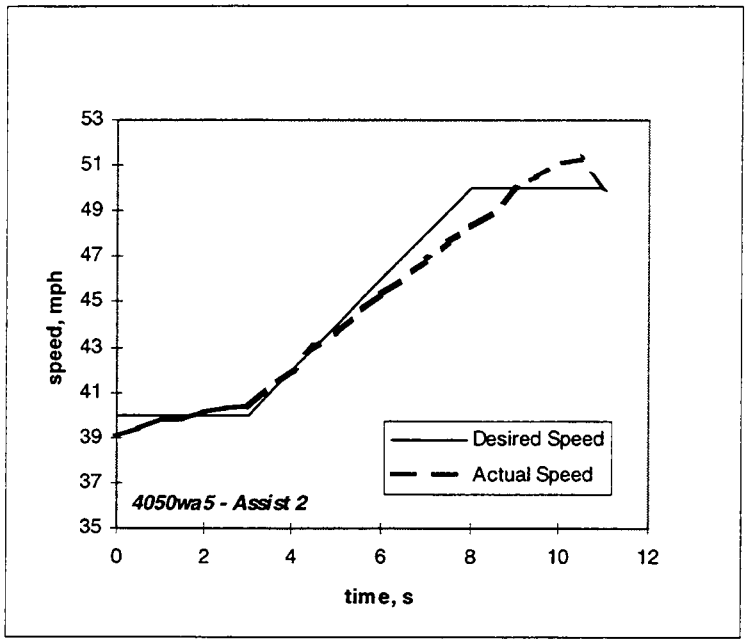


(a) Vehicle Speed Response

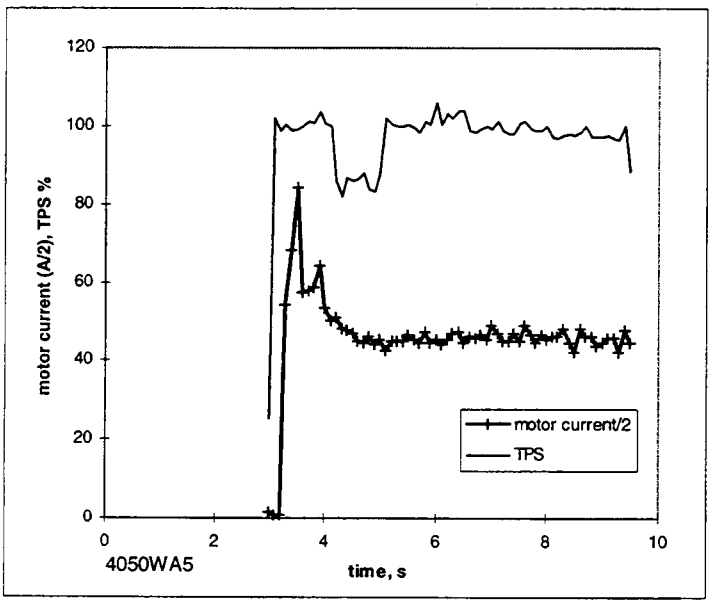


(b) Throttle Position and Motor Current

Figure 40: Acceleration Test #4 Results Using Assist 2 Strategy



(a) Vehicle Speed Response



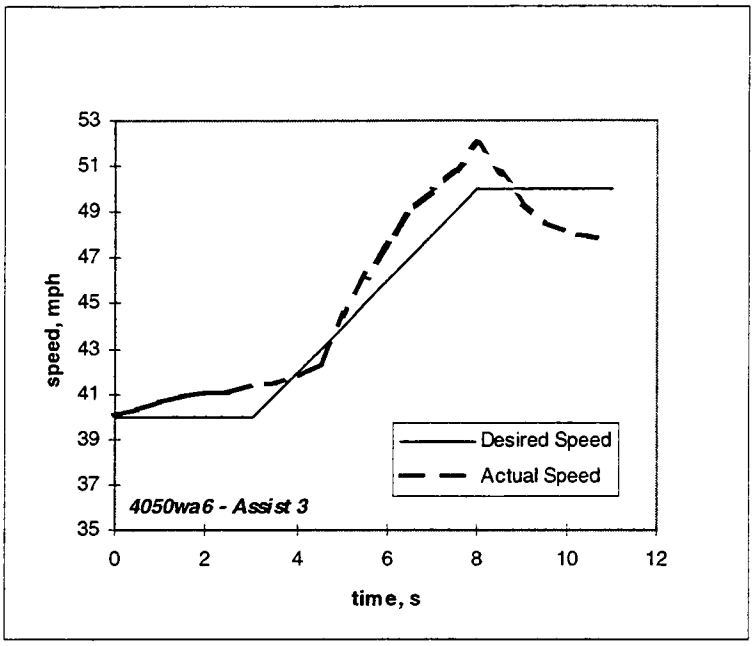
(b) Throttle Position and Motor Current

Figure 41: Acceleration Test #5 Results Using Assist 2 Strategy

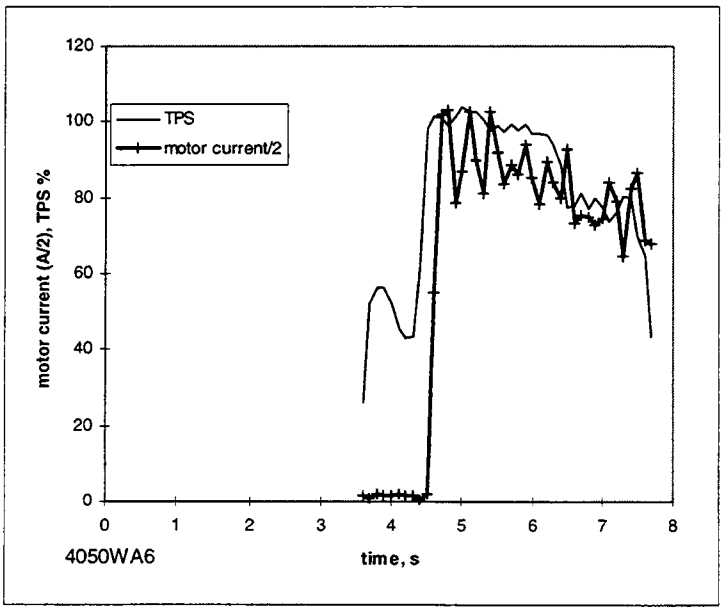
near 180 A. The disadvantage of this method appears in the large fluctuations in motor current (and thus torque). This appeared subjectively as well; the transition to assist did not occur as smoothly using Assist 3 as it did with the other two strategies. It is believed that proper conditioning of the commanded speed signal to the EMC could limit the fluctuation in current which occurs during assist.

The next area of evaluation was deceleration testing. Three test strategies were used; In each, a linear regeneration strategy with differing maximum commanded regeneration levels was used (40%, 60%, and 80% respectively).

Testing proved that using the lowest setting for the maximum regeneration level had almost no effect on vehicle deceleration. Conversely, using the highest level of maximum regeneration proved to be too much of a strain on the engine; getting up the 30 mph starting speed of the deceleration cycle was almost impossible. Therefore, the medium maximum regeneration level proved most effective. Its results compared to the results for no regeneration are shown in Figure 44. The deceleration for no regeneration amounts to approximately 0.4 mph/s; for the three regeneration tests, the deceleration increased to 1.3 mph/s. As shown, the three regeneration test (#1, #2, #3) proved very consistent.

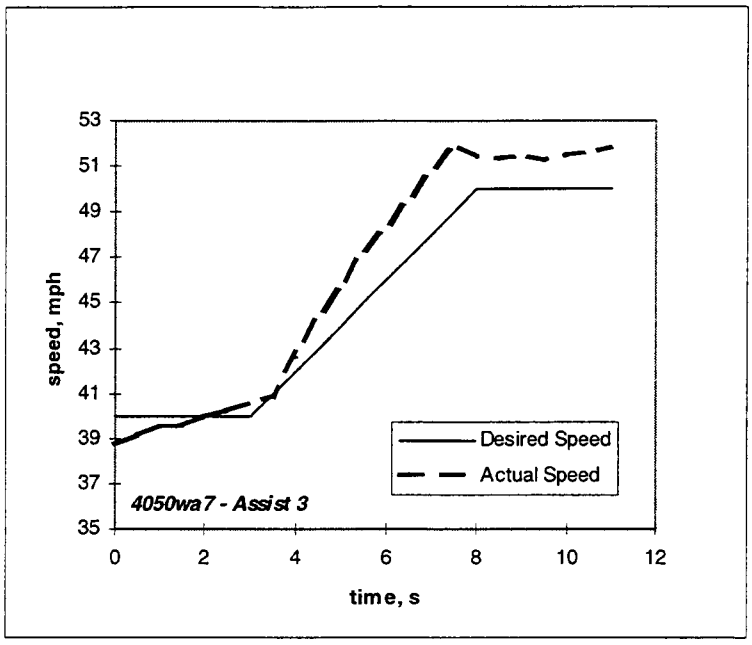


(a) Vehicle Speed Response

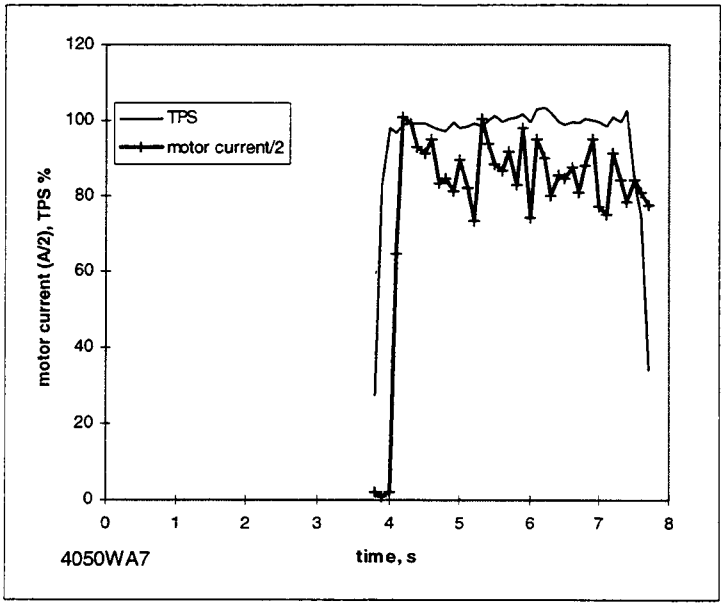


(b) Throttle Position and Motor Current

Figure 42: Acceleration Test #6 Results Using Assist 3 Strategy



(a) Vehicle Speed Response



(b) Throttle Position and Motor Current

Figure 43: Acceleration Test #7 Results Using Assist 3 Strategy

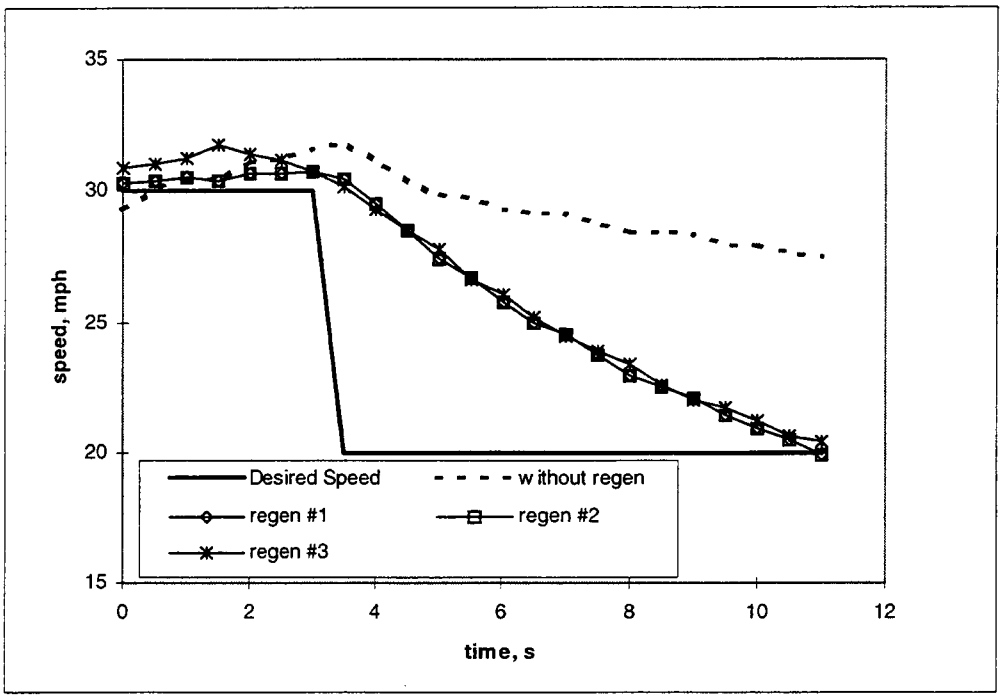


Figure 44: Deceleration Test Response With and Without Regeneration

First gear acceleration tests were conducted to evaluate the effect of the first gear assist. Two problems prevented this testing from providing successful results. First of all, testing showed that the vehicle was perfectly capable of meeting the required acceleration for the driving cycle. This would imply that the first gear assist was not needed at all. Second, testing showed no significant difference in the acceleration of the tests with and without first gear assist. This was later attributed to an intermittent failure in the contact mechanism of the first gear assist switch.

Several conclusions have been drawn based on this data as well as subjective evaluation. First, the vehicle in fact does not require first gear assist except when trying to start on a hill. This was not simulated in the drive cycle used. Second,

first gear assist does help the vehicle under these situations. This evaluation was only made subjectively, however, via road testing.

The final test attempt in the formal vehicle testing involved looking into the response of the regenerative braking system of the vehicle. However, these maximum effort braking tests proved difficult to control on the dynamometer. This area then fell back to subjective evaluation; multiple drivers have confirmed that braking feels solid and correct. Based on this subjective approach, the commanded regeneration level for braking has been set at 80%.

Formal Testing Results - Energy Efficiency Tests

For energy efficiency testing, six total tests (two for each strategy) were conducted using the FUDS cycle to ensure repeatability of the electrical energy consumption data. This data is shown in Table 5.2.

Table 5.2: Energy Efficiency Test: Electrical Energy Consumption

Strategy	Energy Consumed Test 1 (kW-hr)	Energy Consumed Test 2 (kW-hr)	Average Energy Consumed (kW-hr)
Charge-Sustaining	0.04	-0.02	0.01
Charge-Depleting	0.16	0.18	0.17
Ultra Charge-Depleting	0.17	0.13	0.16

In each case, two tests were conducted to indicate repeatability of the results. This is important because driver behavior can strongly influence the results. Therefore, these results are reasonable because they are fairly repeatable.

However, these results would likely be different for different drivers. Since the point of this study is comparison, consistency is the paramount concern.

In the charge-sustaining case, almost no energy is consumed over the driving cycle; in fact, the second test showed an a negative total energy consumption. Both charge-depleting (with regenerative braking) and ultra charge-depleting (without regenerative braking) strategies consume approximately the same amount of energy over the cycle. Several conclusions can be made: first of all, the equality of the two charge-depleting strategies indicates that regenerative braking did not significantly impact energy consumption over the cycle. This may be due to the fact that the dynamometer cannot correctly simulate vehicle inertia during deceleration and especially braking. Therefore, the actual contribution of regenerative braking would be higher on the road than in this test. The next conclusion is that regeneration/charging does produce charge-sustaining operation and in this test contributes all of the electric energy which is recovered from the powertrain.

Furthermore, fuel consumption was also monitored during these tests. The data proved somewhat incongruous; while four of the six data points demonstrated a consistent value near 22 mpge (miles per gallon equivalent), two of the tests posted approximately 40 mpge. This difference is caused by the lack of precision of the pressure gauge; see Appendix D for detail.

Chapter 6

CONCLUSIONS

The success of the project can only be evaluated by comparing the results to the initial objectives. If these objectives have been met, then the project may be deemed successful. In this case, the initial objectives of the study were stated in chapter one. They will be repeated here and then compared against the results in each area.

1. *The final control system should produce a vehicle with driving performance, familiarity, and ease of operation similar to today's conventional cars.* To satisfy this objective, both formal and informal tests have been conducted. Formally, the vehicle has shown its ability to provide smooth response to driver inputs during acceleration and deceleration. Additionally, the vehicle has also shown its ability to successfully follow predetermined driving cycles on the dynamometer which are intended to simulate actual driving conditions. Finally, informal testing on the road has shown the UT-HEV to be capable of providing reasonable performance without any major adjustments in driving style. Based upon these tests, the study has satisfied the first research objective.

2. *The control system should be designed in such a manner as to allow quick, easy changes in control strategies for research evaluation purpose in this study as well as for future studies.* The testing process of this study has itself proven this objective to be met. Many changes have been required during the testing of different control strategies. Through the use of the external settings file (as mentioned in chapter four) and dash-mounted switches (regeneration disable), changes to

the control system have been made quickly and easily. Extensive comments in the control code itself (see Appendix C) provide any future user the ability to quickly understand and modify the program.

3. *The control strategies and settings will be based on a theoretical study augmented by experimental comparisons.* The initial simulation results in chapter three demonstrated the culmination of the modeling process used in the study. These results directed experimentation which proved out the choice of each control strategy based upon the criteria of objective one as well as energy efficiency criteria. As mentioned in the objective statement, this project was not meant to produce the "optimal" control strategies and settings; instead, the final result of this study was simply a well-working control system. Through formal testing and informal drives on the road, the vehicle has been shown to meet this goal.

Testing has shown, to a reasonable extent, that the control system for the vehicle does in fact meet all of the project requirements. Areas of opportunity for further evaluation of the current system as well as for future changes to this system are dealt with in chapter seven.

Chapter 7

RECOMMENDATIONS

Improvements to Current Work

While the current project has shown success in meeting initial project goals, several opportunities to better the system and better its evaluation exist.

Control System

The control system itself currently demonstrates two deficiencies when compared to the requirements document found in the early pages of chapter three. First of all, the battery state of charge function, at the time of this writing, does not function correctly. While implementation of the monitoring algorithm has been made within the control program, the program has been unsuccessful in communicating with the data acquisition system. This serial communications problem is due to deficiencies in the software. The author will remedy this problem by mid-May, 1997. The second area involves the flat panel display. Problems have arisen in the interface between the system controller video card and the display. This hardware problem will also be solved by the author by mid-May.

Evaluation

Because of limitations in both time and equipment, an improved evaluation of the vehicle performance should be made. For comparison purposes, acceleration tests must be carried out and their results compared to those of a production Neon. This will enable the measurement of success in meeting the first research objective. It is extremely important this be carried out also because the

dynamometer testing conducted during formal testing cannot be used to make any conclusive statements about *absolute* vehicle performance (acceleration, etc.) because of the inability to calibrate the equipment properly. The relative conclusions made, however, still stand. Additionally, to provide absolute evaluation of vehicle energy efficiency, the same FUDS cycles should be run at an EPA-certified facility with the control strategies used in this study. These results would better reflect the ability of vehicle inertia (through regenerative braking) to contribute to charging the ESS.

Future Work

The UT-HEV

Future work on the Neon UT-HEV could provide greater insight into hybrid electric vehicle behavior. A true optimization process involving extensive testing could be carried out to determine which control settings are optimal for energy efficiency and performance. Because these control settings are already defined, vehicle performance could be mapped over a wide range of settings and then computer algorithms could then be used to analyze the results and determine optimum settings to improve overall energy efficiency (or emissions, or CNG consumption, etc.) given a base level of vehicle performance.

Improvements to the control system itself might also be worth a second look. The greatest opportunity here lies in the implementation of the same control strategy using the QNX operating system. This would produce smoother execution of the multiple tasks and could possibly increase the overall speed of execution of the program.

Other HEV Projects

The possibility of other HEV work at UT makes this project even more important. It is the author's opinion that the lessons learned in the development

of this control system should be applied to all other control systems used in future HEVs. First of all, the hardware configuration has proven to be reliable, flexible, and simple to install and maintain. Also, the speed of execution afforded by these industrial computers allows the programmer to focus on the code structure and function without having to spend hours optimizing the code for speed. The ability to make quick changes should be considered its largest asset; this is especially important in student competitions where development time is extremely limited.

The basic control strategy used could provide real benefits for any parallel or split-mode hybrid. The author believes that the basic software approach has proven itself to be flexible and provide excellent performance. Development time could be significantly reduced by simply modifying the existing code instead of designing an all-new code.

REFERENCES

References

1. Northeast Sustainable Energy Association, "NESEA Fact Sheets," <http://nesea.nrel.gov/nrel.html#aec>, January 1997.

2. U.S. DOE Hybrid Electric Vehicle Program, "Energy Security and Clean Air," <http://www.hev.doe.gov/general/ontheroad.html>, January 1997.

3. PNGV Team, *Partnership for a New Generation of Vehicles Program Plan*, Washington DC, U.S. Department of Commerce (PNGV Secretariat),, July 1994.

4. Wouk, Victor, "Hybrids: Then and Now," *IEEE Spectrum*, July 1995: 16-21.

5. Argonne National Laboratory (Energy Systems Division), "Competition Results: HEV Challenge Vehicles Exceed Standards in Energy Economy and Emissions," *FutureDrive*, Autumn 1995: 5.

6. Jost, Kevin, "GM's PrEView Impact Electric Vehicle," *Automotive Engineering*, February 1995: 85-89.

7. *Electric Powered Interurban Commuter—Electric Minivans for Fleets.* Auburn Hills MI, Chrysler Corporation, May 1996.

8. McKenna, Michael, "The Myth of 'Zero-Emissions' Vehicles," *National Review*, May 29 1995.

9. U.S. DOE Hybrid Electric Vehicle Program, "Energy Loss," <http://www.hev.doe.gov/general/energyloss.html>, January 1997.

10. Kalberlah, A., "Electric Hybrid Drive Systems for Passenger Cars and Taxis," Society of Automotive Engineers paper 910247, Warrendale PA, 1991.

11. Burke, A. F., "Hybrid/Electric Vehicle Design Options and Evaluations," Society of Automotive Engineers paper 920447, Warrendale PA, 1991.

12. Yamaguchi, Kozo et al. "Development of a New Hybrid System—Dual System" Society of Automotive Engineers paper 960231, Warrendale PA, 1996.

13. Bryce Anderson et al., "1993 Ford Hybrid Electric Vehicle Challenge Final Design Report –The University of Tennessee," *Innovations in Design: 1993 Hybrid Electric Vehicle Challenge (SP-980)*, Warrendale PA, Society of Automotive Engineers, 1994: 229-241.

14. Masding, P. W. and Bumby, J. R. "Integrated Microprocessor Control of a Hybrid I. C. Engine/Battery-Electric Automotive Powertrain" *Transactions of the Institute of Measurement and Control*, Vol. 12 No. 3, 1990: 128-146.

15. 1995 HEV Design Class, "University of Tennessee 1995 Hybrid Electric Vehicle Design," Knoxville TN, University of Tennessee, May 1995.

16. Sluder, Scott, "1996 Hybrid Electric Vehicle Rules and Regulations (version 311/95)," Argonne IL, Argonne National Laboratory: Center for Transportation Research—Energy Systems Division, March 11 1996.

17. Laplante, Phillip A., *Real-Time Systems Design and Analysis*, New York, IEEE Press, 1993.

18. Marr, W. W. "User's Guide to EAGLES version 1.1," Argonne IL, Argonne National Laboratory. Center for Transportation Research—Energy Systems Division, October 1993.

19. Robert Bosch GmbH, *Automotive Handbook*, 3rd Ed, Warrendale PA, Society of Automotive Engineers, 1993: 325.

20. Korff, Walter H. *Designing Tomorrow's Cars*, Burbank CA, M-C Publications, 1980: 50-51.

21. Cole, G. H. "SIMPLEV: A Simple Electric Vehicle Simulation Program Version 2.0," Idaho Fall ID, U.S. DOE (DOE/ID-10293-2), April 1993.

22. Advanced Research Projects Agency Electric and Hybrid Vehicle Technical Program, "Hybrid Electric Vehicle Simulink Toolbox—User's Guide," <http://www.ev.hawaii.edu/Simulation/HEV_ARPA/hevbox.html>, January 1997.

23. Masding, P. W. and Bumby, J. R. "Identification and Performance Simulation of a Hybrid I. C. Engine/Battery Electric Automotive Power Train Part I: The I. C. Engine," *Transactions of the Institute of Measurement and Control*, Vol. 12 No. 1, 1990: 27-39.

24. Morris, R. L., H.G. Hopkins, R. H. Borcherts, "An Identification Approach to Throttle-Torque Modeling," Society of Automotive Engineers paper 810448, Warrendale PA, 1981.

25. Krause, Paul C. and Wasynczuk, Oleg, Electromechanical Motion Devices, New York, McGraw-Hill, 1989: 273-327.

26. Unique Mobility, "SR180P/CR20-300 Data Sheet," Golden CO, October 1994.

27. Reliance Motion Control, *DC Motors, Speed Controls, Servo Systems—the Electro-Craft Engineering Handbook*, Eden Prairie MN: 4-31, 4-43.

28. Unique Mobility, "Installation and Operating Instructions: SR180 Brushless DC Motor, CR20-300A Brushless DC Motor Controller", Golden CO, December 1994.

29. Munger, Frank, "Plug-in Transportation," *Knoxville News Sentinel*, February 24, 1997, final ed., sec. B: 1.

30. Octagon Systems, "Micro-PC 1995 Catalog," Westminster CO, 1995: 22-25.

31. Wilkerson, Jordan et al. "The University of Maryland at College Park Methanol Hybrid Electric Vehicle," *1994 Hybrid Electric Vehicle Challenge (SP-1103)*, Warrendale PA, Society of Automotive Engineers, 1994: 181-193.

32. Quintero, Richard and Barbera A. J., "A Real-Time Control System Methodology for Developing Intelligent Control Systems," National Institute of Standards and Technology, Robot Systems Division, Gaithersburg MD, 1992.

33. Edwards, Keith, *Real-Time Structured Methods*, New York, Wiley, 1993.

34. Lampley, Stephen R., *The Evaluation of a Dedicated Natural Gas Vehicle Conversion*, M.S. Thesis, Knoxville TN, University of Tennessee, May 1995.

35. Malcosky, Norman and Thompson, David, *COMPRES* Computer Software. Columbia Gas System Service Corporate Research Development, 1991. MS-DOS executable (QBASIC source code), 69 KB.

APPENDICES

Appendix A

SIMULATION PROGRAM: CONSTRUCTION, ASSUMPTION, AND PARAMETERS

The simulation program used in this study was executed using Microsoft EXCEL™ 7.0. The program used multiple worksheets to separate basic parameters required for final calculations and display of results:

- Results: display final vehicle speeds, throttle position, and component power output
- Raw Data: contains engine modeling data used for HPU calculations
- Vehicle Parameters: contains all physical parameters associated with the vehicle (mass, etc.)
- Drive Cycle: contains the specified driving cycle for the test
- Controls: vehicle controls settings for assist and driver response
- Calculations: all raw calculations are done here

In the simulations, several assumptions are made:

1. Dynamic effects (speed feedback) may be ignored for HPU model. Therefore, HPU output power is a function of engine speed and throttle position alone (using steady-state values).
2. EM dynamic effects are substituted by a first-order lag.
3. No additional time delay is incurred during gear shifting. Additionally, the minimum and maximum engine speeds (for gear shifting) are 1500 RPM and 3500 RPM respectively.
4. The driver may be modeled as a speed-feedback proportional controller as described in chapter three.

The following parameters indicate the set used for the entire simulation results included in this study.

Engine Power - Final Curve Fit

A		B		C	
a	-1.7377	c	84.9953	e	44.60706
b	0.005199	d	-0.054003	f	-0.142087
				g	5.69E-05

Final Equation: $P = a + b\,(RPM) + [\,c + d\,(RPM)\,] / (TPS) + [\,e + f\,(RPM) + g\,(RPM)^2\,] / (TPS)^2$

General Specifications

drag coefficient	0.328	final drive ratio	3.55
frontal area	20.6 sq. ft.	1st gear	3.54
rolling resistance	0.007	2nd gear	2.12
weight	3375 lbm	3rd gear	1.36
rolling radius	0.9799 ft	4th gear	1.03
drivetrain efficiency	0.95	5th gear	0.72

Engine Characteristics

Assumed power (max) at 3000 rpm	35 hp
Required scaling factor (R)	2.819625

Engine Power / TPS relation:

Power = R { a + b (RPM) + [c + d (RPM)] / (TPS) + [e + f (RPM) + g (RPM)^2] / (TPS)^2 }

Parameters

R	2.819625	d	-0.054
a	-1.7377	e	44.60706
b	0.005199	f	-0.14209
c	84.9953	g	5.69E-05

General Controls

Threshold TPS_hi	25 %	Threshold TPS_lo	25 %
Threshold TPS_hi2	100 %	Max. regen current	10 A 0
K - driver	10 %TPS/mph		

Assist Control

If TPS > TPS_hi:

*I_desired = 200 A / (TPS_hi2 - TPS_hi) * (TPS - TPS_hi)*

IF TPS < TPS_hi:

I_desired = 0

Motor Model

Speed-I const., K_iv		0.0025 V/A
Minimum RPM	HPU	1500 RPM
Maximum RPM	HPU	3500 RPM

Note: calculation time interval = 0.1 s.

Appendix B

SYSTEM CONTROLLER HARDWARE AND ASSOCIATED EQUIPMENT

This appendix is merely meant to provide additional detail about the components of the system controller and their specifications, source, and price.

System Configuration:

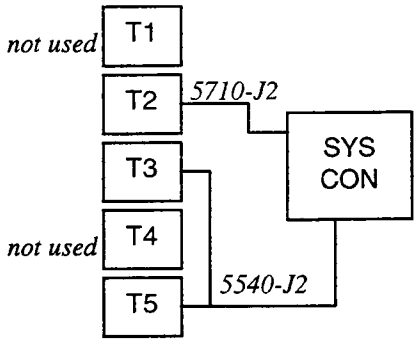
Controller

Description	Part Number	Manufacturer	Supplier	Price
CPU	5025A	Octagon	Marshalls	$1,164.00
Analog I/O	5710	Octagon	Marshalls	$ 370.00
Digital I/O	5540	Octagon	Marshalls	$ 239.00
Storage	5815	Octagon	Marshalls	$ 248.00
Video	5420	Octagon	Marshalls	$ 248.00
Card Cage	5207RMH	Octagon	Marshalls	$ 185.00
Communications (Ethernet)	5500	Octagon	Marshalls	$ 245.00
Power Supply	NFC25-12T05-12	Computer Products	Milgray	$82.00

Other Parts

Description	Part Number	Manufacturer	Supplier	Price
Flat Panel Display	LM32K07	Sharp	Milgray	$ 350.00
Flat Panel Inverter	LS340	Xentek	Milgray	$ 40.00
Terminal Block - 26 pin (2)	STB26	Octagon	Marshalls	$ 75.00
Terminal Block - 20 pin (3)	FLKM 20	Phoenix Contact	Action Elec.	$ 80.00
Opto-Isolation board	G4PB24	Opto 22	Carlton-Bates	$ 115.00
Opto-Isolation Modules (15)	G4IDC5	Opto 22	Carlton-Bates	$ 100.00
Total System Cost				$3,541.00

System Controller Configuration - terminal blocks and wiring interfaces



T2 INPUTS

Junction	Port	Wire	Description	Other
T2-4	GND	-	Ground	
T2-7	5710-IN 3 -	C400	MAP	
T2-8	GND	- C70J	Ground Direction signal from EMC	
T2-11	5710-IN 5 -	C401	TPS	
T2-12	GND	-	Ground	
T2-15	5710-IN 7 -	CC00	Brake pressure (voltage divider)	
T2-16	GND	- C70B	Ground EMC signal ground	
T2-17	5710-DAC0	C07D	EMC commanded speed signal	
T2-19	5710-DAC1	C07E	EMC commanded regen level	

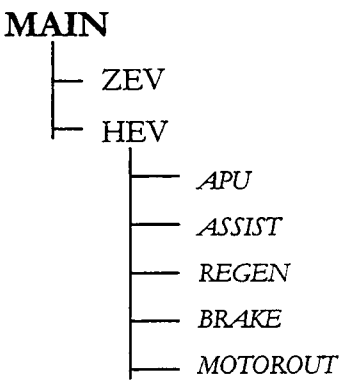
T3 INPUTS

Junction	Port	Wire	Description	Other
T3-1	5540-PA-4	C70K	Motor Temperature from EMC	
T3-3	5540-PA-2	CB00	Regen enable/disable	
T3-5	5540-PA-5	C70L	EMC temperature	
T3-7	5540-PA-1	CB01	HEV / ZEV switch	
T3-11	5540-PA-6	CB02	Ignition	
T3-13	5540-PA-7	CC01	First Gear Switch	
T3-21	NA		+5 V	
T3-23	NA		- 5 V	
T3-24	NA	G0S0	GND	

Appendix C

CONTROL SYSTEM CODE

This appendix presents the complete listing of the control system code. This code was written by Brian E. Tucker and James Hatmaker using Borland Turbo C++ version 3.0 for DOS. The program files ("*.cpp") and the header files ("*.h") are organized as follows.



Each of the program and header files follows.

MAIN.CPP

```
#include <stdio.h>
#include <dos.h>
#include <stdlib.h>
#include "srlport.h"
#include "zev.h"
#include "hev.h"
#include "main.h"

/*
*       FILE:           main.cpp
*
*       PURPOSE:        This file is the entry point for the University of
*                       Tennessee Hybrid Electric Vehicle Powertrain conrols
*                       program.  "main.cpp" accomplishes two main goals:
*                       First, this file sets up the necessary settings,
*                       initializations, and constants needed to carry out
*                       the other functions.  Second, this part of the
*                       controls program determines the driver-selected mode
*                       (either HEV or ZEV) and calls the corresponding
*                       function.
*
*       PROCEDURE:      Step 1:  initialize data values
*                       Step 2:  port initialization
*                       Step 3:  read settings file
*                       Step 4:  read serial port
*                       Step 5:  mode check
*                       Step 6:  call appropriate mode function
*
*
*       LINEAGE:        Brian E. Tucker, James Hatmaker         1.00   5/5/96
*                               - original version
*
*                       Brian E. Tucker                 1.01    10/27/96
*                               - added comment blocks
*
*                       Brian E. Tucker                 1.02    1/27/97
*                               - additional comment blocks
*
*
*/

/*  Create data / settings structures  */

DATASTRUCT data;                /* global data from serial port */
SETTINGSSTRUCT setting;         /* settings (default or from file) */
SerialPort serialPort;          /* create instance of SerialPort class */

main()
{

/*
*  STEP 1:  INITIALIZE DATA VALUES
*   This section sets up the default values of data and settings
*   structures Theses values will be replaced with real data from serial port
*   (data struct), and external file info.(setting struct).
*/

/* data structure defaults - info. from Cruising Equipment Kilowatt Hour
   meter  (KWHR meter) supplied for competition purposes */

        data.time       = 1;        /* time number */
        data.batKWHRS   = -0.02;    /* kwhr's used */
        data.batCurrent = 12.4;     /* current being drawn from battery */
```

```
        data.batVolt    = 220.0;    /* battery pack voltage */
        data.batAHRS    = -0.05;    /* amp-hours used */
        data.CNGPress   = 3000;     /* pressure in CNG tank */
        data.CNGTemp    = 27;       /* CNG tank temperature */

/* setting structure defaults - program parameters.  Can be set in
   "settings.ini" file external to program */


        // floats
        setting.batt_capacity          = 17.2; /* battery capacity, amp-h */

   /* for many of the following values, a 0-100 scale has been used.  In each
      case, 0 corresponds to 0 V and 100 corresponds to 5 V at the ADC.
      This is abbreviated V*20 (volts * 20.0; e.g. 5V * 20 = 100)
      Otherwise, if the value is labeled "%", this corresponds to a percentage
      of full scale (e.g., TPS = 100 % => TPS = TPS_max V*20) */

        setting.BPRESS_min      = 10.0; /* minimum brake pressure
                                           to activate regen0, V*20 */
        setting.BPRESS_max      = 90.0; /* maximum brake pressure
                                           to activate regen, V*20  */
        setting.TPS_min = 13.0; /* minimum possible TPS, V*20 */
        setting.TPS_max = 64.0; /* maximum possible TPS, V*20 */

        // int
        setting.SOC_min = 60;   /* minimum SOC allowed; used
                                           for regen logic, %       */
        setting.SOC_max = 95;   /* maximum SOC allowed      */
        setting.TPS_HInorm      = 50;   /* High TPS under normal
                                           conditions; for assist, % */
        setting.TPS_LOnorm      = 0;    /* Low TPS under normal
                                           conditions; for assist, % */
        setting.TPS_HIdicta     = 35;   /* High TPS under dictator
                                           assist, now defunct, %  */
        setting.TPS_LOdicta     = 0;    /* Low TPS under dictator
                                           assist, now defunct, %   */
        setting.TPS_HIdictr     = 60;   /* High TPS under dictator
                                           regen; now defunct, %    */
        setting.TPS_LOdictr     = 40;   /* Low TPS under dictator
                                           regen; now defunct, %    */
        setting.TPS_1G          = 20;   /* Low TPS in first gear, % */
        setting.AVG_CYCLE       = 5;    /* Digital filter parameter */



/*
*       STEP 2:  PORT INITIALIZATION
*       Setup the parameters of the analog (5710) and digital (5540)
*       I/O cards.
*/

 /*     5710: Port A,B,C = output
        from table on page 37 of the 5710 manual */

        outportb(BASE_5710+0x0B, 0x80);

 /*     5540: Port A,B,UC = input; Port LC = output
        from Table 6.1. p.23 of the 5540 manual         */

        outportb(CTRL_5540, 0x9A);



/*
*       STEP 3:  READ SETTINGS FILE
```

```
*       In this section, we read the "settings.ini" external file
*       and place this data into the setting structure.  The format
*       of this data corresponds to the default values listed for the
*       settings structure previously listed.
*/

        FILE *stream;
        char buf[60];

        stream = fopen("SETTINGS.INI", "r");
        if(!stream)
        {
           printf("Cannot open Settings.INI default settings file.\n");

            /* In this case, the file cannot be opened.
               Set the parameters to their default since the
               file does not exist */

        }
        else
        {
                /* seek to the beginning of the file */
                fseek(stream, 0, 0);

                /* read the data and display it  */
                fread(buf, 1, 55, stream);

                fclose(stream);

                int bufcount = 0;       /* counter for buffer position */
                float tmpfloat[5];       /* temp. storage for floats */
                int tmpint[10];          /* temp. storage for ints */

                        /* loop number of new lines in the file  */
                for(int i = 0; i<15; i++)
                {
                        /* reads chars from the file until it reaches EOF
                           and store in an array called tmpstr */

                        int j = 0;
                        char tmpstr[10];
                        do
                        {
                                tmpstr[j] = buf[bufcount+j];
                                j++;
                        }
                        while (buf[bufcount+j] != 0x0A);

                        j++;
                        bufcount += j;

                        tmpstr[j-1] = 0;

                        if(i<5)
                                // converts first five values to floats
                                tmpfloat[i] = atof(tmpstr);

                        else
                                // converts all others to integers
                                tmpint[i-5] = atoi(tmpstr);

                }
                // store values in the setting stucture

                // floats
                setting.batt_capacity          = tmpfloat[0];
                setting.BPRESS_min             = tmpfloat[1];
```

```
                setting.BPRESS_max              = tmpfloat[2];
                setting.TPS_min         = tmpfloat[3];
                setting.TPS_max         = tmpfloat[4];

                // int
                setting.SOC_min         = tmpint[0];
                setting.SOC_max         = tmpint[1];
                setting.TPS_HInorm              = tmpint[2];
                setting.TPS_LOnorm              = tmpint[3];
                setting.TPS_HIdicta             = tmpint[4];
                setting.TPS_LOdicta             = tmpint[5];
                setting.TPS_HIdictr             = tmpint[6];
                setting.TPS_LOdictr             = tmpint[7];
                setting.TPS_1G                  = tmpint[8];
                setting.AVG_CYCLE               = tmpint[9];

        }

/*
*       STEP 4:  READ SERIAL PORT
*       Here, we create the com buffer, test the port, and read
*       the data
*/

                /* create a buffer for the serial port input */
        if(serialPort.CreateBuffer())
        {
                printf("created and flushed the COM buffer\n");
        }


/*      for(long a = 0; a < 1000000; a++)
        {
                serialPort.testPort();
                ReadSerial();
        }
*/

        for(int ab=0; ab < 30; ab++)
                ReadSerial();
        return 1;



/*
*       STEP 5:  MODE CHECK
*       Check the HEV/ZEV switch to determine mode.
*       HEV/ZEV switch - 5540 PORT A1
*       tmpo37 =1 = HEV
*       tmpo37 =0 = ZEV
*       (tmpo37 - storage for value read from port)
*/

        int tmpo37 = inportb(PORTA_5540);  /* read port */
        tmpo37 &= P1;                      /* check bit 1 (HEV/ZEV) */
        MODESTRUCT mode;
        if(tmpo37)
                mode = HEV;
        else
                mode = ZEV;


        if(mode == HEV)
                printf("HEV mode");
        else
                printf("ZEV mode");
```

```
/*      STEP 6:  CALL APPROPRIATE MODE FUNCTION
*       Create instances of mode classes and call functions
*/

        ZElectricV zev;
        HElectricV hev;


        if(mode == ZEV)
                zev.EnterPoint();
        else
                hev.EnterPoint();



        return 1;
}



DATASTRUCT* getData(void)
{
        return &data;
}

SETTINGSSTRUCT* getSettings(void)
{
        return &setting;
}


void ReadSerial(void)
{
        char display[200];
        SDATASTRUCT* slastdata = serialPort.getData();

        data.time              = atoi(slastdata->time);
        data.batKWHRS   = atof(slastdata->sbatKWHRS);
        data.batCurrent = atof(slastdata->sbatCurrent);
        data.batVolt    = atof(slastdata->sbatVolt);
        data.batAHRS    = atof(slastdata->sbatAHRS);
        data.CNGPress   = atoi(slastdata->sCNGPress);
        data.CNGTemp    = atoi(slastdata->sCNGTemp);

        sprintf(display,"%d, %f, %f, %f, %f, %d, %d\n", data.time,
                data.batKWHRS, data.batCurrent, data.batVolt, data.batAHRS,
                data.CNGPress, data.CNGTemp);
        printf(display);
        return;
}
```

MAIN.H

```
// Main.h
// Header for the main

#ifndef m_main
#define m_main

// define card and port addresses

#define BASE_5540 0x0120       /* set to 120H (jumper setting) */
#define BASE_5710 0x0100       /* set to 100H (default) */
#define BASE_CTC  4+BASE_5710
#define BASE_8255 8+BASE_5710
#define CH_ADDR   9+BASE_5710

#define PORTA_5540      BASE_5540
#define PORTB_5540      BASE_5540 + 0x01
#define PORTC_5540      BASE_5540 + 0x02
#define CTRL_5540       BASE_5540 + 0x03

#define PORTA_5710      BASE_5710 + 0x08
#define PORTB_5710      BASE_5710 + 0x09
#define PORTC_5710      BASE_5710 + 0x0A
#define CTRL_5710       BASE_5710 + 0x0B
#define CTC2_5710       BASE_5710 + 0x06
#define CNTCTL_5710      BASE_5710 + 0x07
#define WATCHDOG        0x201
#define ON              0x01

// defines for analog channel functions
#define CH_V_ACT        0x01    /* actual speed on ADC channel 1 */
#define CH_MAP          0x03    /* MAP on ADC channel 3 */
#define CH_TPS          0x05    /* TPS on ADC channel 5 */
#define CH_I_MOTOR      0x06    /* motor current on ADC channel 6 */
#define CH_BPRESS       0x07    /* brake press. on ADC channel 7 */

// defines for port calls
#define P0              0x01
#define P1              0x02
#define P2              0x04
#define P3              0x08
#define P4              0x10
#define P5              0x20
#define P6              0x40
#define P7              0x80


// data structure:  information from the Data Acquisition System (KWHR meter)

typedef struct
{
        int     time;
        float   batKWHRS;
        float   batCurrent;
        float   batVolt;
        float   batAHRS;
        int             CNGPress;
        int             CNGTemp;
}DATASTRUCT;

// mode structure (HEV or ZEV)

typedef enum
{
```

```
        HEV,
        ZEV
}MODESTRUCT;

//settings structure:  read from external file (or use default)

typedef struct
{
        float batt_capacity;
        float BPRESS_min;
        float BPRESS_max;
        float TPS_min;
        float TPS_max;
        int   SOC_min;
        int   SOC_max;
        int   TPS_HInorm;
        int   TPS_LOnorm;
        int   TPS_HIdicta;
        int   TPS_LOdicta;
        int   TPS_HIdictr;
        int   TPS_LOdictr;
        int   TPS_1G;
        int       AVG_CYCLE;
}SETTINGSSTRUCT;

// function prototypes

DATASTRUCT* getData(void);
SETTINGSSTRUCT* getSettings(void);
void ReadSerial(void);


#endif
```

ZEV.CPP

```
// zev.cpp
#include <dos.h>
#include "zev.h"
#include "main.h"

/*
*       FILE:           zev.cpp
*
*       PURPOSE:         "zev.cpp", once called from "main.cpp" directs
*                       control of the Unique Mobility SR-180 & CR-20/300A
*                       to propel the vehicle.  The powertrain control
*                       consists of reading driver inputs (TPS, brake) and
*                       calculating needed outputs (motor commanded speed
*                       and regeneration) to send to the motor.
*
*       PROCEDURE:      Step 1:  initialization of variables
*                       (Main Loop)
*                       Step 2:  monitor and display system state
*                       Step 3:  read driver input
*                       Step 4:  filter throttle position signal (TPS)
*                       Step 5:  calculate motor output
*                       Step 6:  output to motor
*
*
*       LINEAGE:        Brian E. Tucker, James Hatmaker         1.00   5/5/96
*                               - original version
*
*                       Brian E. Tucker                 1.01    10/27/96
*                               - added comment blocks
*
*                       Brian E. Tucker                 1.02    1/28/97
*                               - additional comment blocks
*
*                       Brian E. Tucker                 1.10    future date ?
*                               - changed TPS from V*20 to %
*                               - added function ADconv(int) to consolidate
*                                 analog to digital conversion functions
*
*
*/


ZElectricV::ZElectricV(void)
{

        /* for safety, ensure that not TPS values are accepted for the first 3+
sec. */

        ArrayCount = 0;
        for(int i=0; i<50; i++)
                InputTPS[i] = 0;


        curSOC = 75.0;
}


ZElectricV::~ZElectricV(void)
{
}

int ZElectricV::EnterPoint(void)
{
```

```
/*
*  STEP 1: INITIALIZATION OF VARIABLES
*     Here, we simply read in the settings values from the external file
"settings.ini"
*   as well as initialize other variables needed for the main loop.
*/

        SETTINGSSTRUCT* set = getSettings();   /* get settings information */

        int index;
        int tempBPRESS = 0;
        int a = 0;                             /* counter for infinite loop */
        preTime = time(0);                     /* time counter for EverySec */


/***** MAIN LOOP *****/

        while(a <= 1000)
        {

/*
*  STEP 2: MONITOR AND DISPLAY SYSTEM STATE
*   The first function here, EverySec, runs its member functions only one time
*    per second.  These functions, which include updating the display, safety
checks,
*   etc., are considered less important than the powertrain control.  Next, the
*   watchdog timer is reset to ensure that the system will remain operating.
*/

                EverySec();                            /* run once per second */

                outportb(WATCHDOG,ON);         /* reset watchdog timer */

/*
*  STEP 3: READ DRIVER INPUT
*    This operation involves reading values of brake, throttle, etc. from the
driver.
*/

                getDriverInput();                      /* read brake, TPS */

/*
*  STEP 3A:  MONITOR MOTOR STATE
*       This section has been added to monitor motor current and actual
*   speed signals.  It may be commented out for high speed operation.
*/

        // get actual speed

        outportb(CH_ADDR, CH_V_ACT);
        delay(1);
        outportb(BASE_5710, 0xFF);

        int tmp4 = inportb(BASE_5710);
        tmp4 &= 0x01;
        while(!tmp4)
        {
                tmp4 = inportb(BASE_5710);
                tmp4 &= 0x01;
        }

        // note: this value of V_ACT ranges from 2048-0 (i.e., 0 to -5 V )
        float curV_ACT = (inportb(BASE_5710+2)*16 + inportb(BASE_5710+3)/16);
        // convert to range of 0 ~ 100%
        curV_ACT = -2.0 * 100.0 / 2048.0 * (curV_ACT - 2048.0); // 0 < curV_ACT
< 100
```

```
        // get motor current


        outportb(CH_ADDR, CH_I_MOTOR);
        delay(1);
        outportb(BASE_5710, 0xFF);

        int tmp3 = inportb(BASE_5710);
        tmp3 &= 0x01;
        while(!tmp3)
        {
                tmp3 = inportb(BASE_5710);
                tmp3 &= 0x01;
        }

        // note: this value ranges from 2048-0 (i.e., 0 to 5 V )
        float curI_MOT = (inportb(BASE_5710+2)*16 + inportb(BASE_5710+3)/16);
        // convert to range of 0 - 100%
        curI_MOT = 1.0 * 100.0 / 2048.0 * (curI_MOT - 2048.0); // 0 < curI_MOT
< 100

        // now, convert % into amps.  If voltage divider cuts in in half, then
        // 2 V = 100 A (instead of 4 V in original 10 V signal).
        // Therefore, since 2 V = 40% = 100 A, multiply % by 2.5 to get A

        curI_MOT *= 2.5/2;

        // print out values
        printf("vact,%f,imo,%f,",curV_ACT,curI_MOT);



/*
*  STEP 4:  FILTER THROTTLE POSITION SIGNAL (TPS)
*   This algorithm limits the maximum change in TPS and thus reduces
*   noise in the system.  Response may be delayed.
*   Setting "AVG_CYCLE" determines severity of filter.
*
*
*/

                InputTPS[ArrayCount] = curTPS;          /* store TPS in array */
                ArrayCount++;


                if(ArrayCount == set->AVG_CYCLE)        /*    store    "AVG_CYCLE"
values */
                        ArrayCount = 0;


                float tempTPS = 0.0;

                for(int i=0; i<set->AVG_CYCLE; i++)
                        tempTPS += InputTPS[i];/* tempTPS = sum of previous */

                curTPS = float(tempTPS/(float)set->AVG_CYCLE);
                                                        /* final value is average
*/

/*
*  STEP 5:  CALCULATE MOTOR OUTPUT
*   Based upon driver input, this function calculates the necessary values of
*   commanded speed and regeneration level to send back to the motor controller
*/
```

```
                calcMotorOutput();      /* calculate values of COM_V and
                                           REGEN to send to motor */

/*  STEP 6:  OUTPUT TO MOTOR
*    This function performs the conversions necessary to carry out the digital
*    analog conversions required to send a commanded speed and regeneration
*    signal to the motor controller
*/

                OutputToMotor();        /* output COM_V / REGEN values to
                                           motor controller */


                a = a;                  /* ensure loop is infinite */
        }

        return 0;
}


void ZElectricV::EverySec(void)
{

/*
*       EverySec(void)
*
*       PURPOSE:        Runs member functions once every second.  These
*                       functions are of lower priority than the power-
*                       train control functions and are arbitrarily
*                       executed only once a second.  These include
*                       reading data from serial port buffer, calculating
*                       battery state of charge, performing safety checks,
*                       and updating display.
*
*
*       LINEAGE:        Brian E. Tucker, James Hatmaker         1.00   5/5/96
*                               - original version
*
*                       Brian E. Tucker                 1.01   10/27/96
*                               - added comment blocks
*
*
*/


/*  Check time to see if 1 second has elapsed; return if not */

        time_t curtime = time(0);
        if((curtime - preTime) < 1)
                return;

        preTime = curtime;

/*  Perform member functions - only once per second */

        ReadSerial();           /* reads the buffer for the serial port */
        curSOC = calcSOC();     /* calculates battery state of charge */
        SAFETYSTRUCT safety = safetyCheck();    /* safety check */
        Display();              /* updates display */
        printf("second \n");
}
```

```
float ZElectricV::calcSOC()
{

/*
*       calcSOC(void)
*
*       PURPOSE:        Calculates state of charge (SOC) of the battery pack.
*                       The total used amp hours (read from KWHR meter) in the
*                       data structure are added (since this value is < 0) to
*                       the battery pack capacity as entered into the
settings.ini
*                       file.  The resulting number (amp-hours of capacity
*                       available) is then divided by capacity to determine
*                       state of charge (in percent).
*
*
*       LINEAGE:        Brian E. Tucker, James Hatmaker         1.00   5/5/96
*                               - original version
*
*                       Brian E. Tucker                 1.01   1/28/97
*                               - added comment blocks
*
*
*/

        /* get "amp-hours used" (p_data->batAHRS) for KWHR meter */

        DATASTRUCT* p_data = getData();

        /* get "battery capacity" (p_set->batt_capacity) from settings.ini file
*/

        SETTINGSSTRUCT* p_set = getSettings();

        /* calculate state of charge */

        float   SOC   =   (p_data->batAHRS   +   p_set->batt_capacity)   /   p_set-
>batt_capacity * 100;

        return SOC;
}


void ZElectricV::getDriverInput()
{

/*
*       getDriverInput()
*
*       PURPOSE:        To perform A/D conversions to obtain throttle
*                       position and brake pressure
*
*
*       LINEAGE:        Brian E. Tucker, James Hatmaker         1.00   5/5/96
*                               - original version
*
*                       Brian E. Tucker                 1.01   10/27/96
*                               - added comment blocks
*
*
*/


/*  Get throttle position */
```

```
        outportb(CH_ADDR, CH_TPS);      /* select A/D channel       */
        delay(1);                       /* wait 1 ms for conversion */
        outport(BASE_5710, 0xFF);       /* initiates conversion     */


        int tmp = inportb(BASE_5710);  /* wait for end of conversion */
        tmp &= 0x01;
        while(!tmp)
        {
                tmp = inportb(BASE_5710);
                tmp &= 0x01;
        }


        int highThrottle = inportb(BASE_5710+2);
        int lowThrottle = inportb(BASE_5710+3);

        if(highThrottle < 128)
                highThrottle = 128;

        /* note: this value of TPS ranges 2048-4096 */

        curTPS = (float)((float)highThrottle*16.0 + (float)lowThrottle/16.0);


        /* convert to range of 0-100%  */

        curTPS = (curTPS - 2048.0)/20.47; // 0 < curTPS < 100


/* Get brake pressure */

        outportb(CH_ADDR, CH_BPRESS);   /* select A/D channel       */
        delay(1);                       /* wait 1 ms for conversion */
        outport(BASE_5710, 0xFF);       /* initiates conversion     */

        tmp = inportb(BASE_5710);       /* wait for end of conversion */
        tmp &= 0x01;
        while(!tmp)
        {
                tmp = inportb(BASE_5710);
                tmp &= 0x01;
        }


        int highBrake = inportb(BASE_5710+2);
        int lowBrake  = inportb(BASE_5710+3);

        if(highBrake < 128)
                highBrake = 128;

        /* note: this value of brake pressure ranges 2048-4096 */

        curBPRESS = (int)(highBrake*16 + lowBrake/16);

        /* convert to range of 0-100% */

        curBPRESS = (int)((curBPRESS - 2048)/20.47); // 0 < curBPRESS < 100

        return;
}


void ZElectricV::calcMotorOutput()
```

```
{

/*
*       calcMotorOutput()
*
*       PURPOSE:        To calculate values of commanded speed and regen
*                       to send to the motor based upon inputs.
*
*       LINEAGE:        Brian E. Tucker, James Hatmaker         1.00   5/5/96
*                               - original version
*
*                       Brian E. Tucker                 1.01    10/27/96
*                               - added comment blocks
*
*
*/
        SETTINGSSTRUCT* p_set = getSettings();

        // check brake pressure first
        if (curBPRESS > p_set->BPRESS_min)
        {
                // *** At this state, regenerative braking is enabled ***

                  curREGEN = 80;

                // since we're braking, no need for commanded speed

                curCOM_V = 0.0;
        }

        else

        {
                // *** At this state, we're not braking; check throttle ***


                // constant regen level
                curREGEN = 50.0;

                // check TPS for safety, level
                if (curTPS > p_set->TPS_min && curTPS < p_set->TPS_max)

                {
                 // safe condition:  TPS within normal limits
                 // calculate commanded speed based on TPS; no filtering

                 curCOM_V = (float)((((float)curTPS - p_set->TPS_min) * 100.0 /
(p_set-
                 >TPS_max - p_set->TPS_min);

                }

                else
                {
                        // unsafe condition!  TPS sensor has failed!!!
                        curCOM_V = 0;
                }
         }
}



void ZElectricV::OutputToMotor()
{

/*
```

```
*       OutputToMotor()
*
*       PURPOSE:        To perform D/A conversions to output values of
*                       commanded speed and regeneration to motor
*
*
*       LINEAGE:        Brian E. Tucker, James Hatmaker         1.00   5/5/96
*                               - original version
*
*                       Brian E. Tucker                 1.01    10/27/96
*                               - added comment blocks
*
*
*/
        // convert regen, com_v from 0-100 to 0 - 4095
        int COM_V = (int)(curCOM_V * 20.47) + 2048;
        int tmp = inportb(PORTA_5540);


        tmp &= P2;

        int REGEN;
        if (!tmp)
               REGEN = (int)(-1.0 * curREGEN)* 20.47 + 2048;
        else
               REGEN = 0x0800;


        //convert values to two registers needed for analog output
        int tempCOM_V = COM_V;
        int tempREGEN = REGEN;


        // output commanded speed:  channel 1
        outportb (BASE_5710+12, (tempCOM_V % 256));
        tempCOM_V = tempCOM_V >> 8;
        outportb (BASE_5710+13, (unsigned char)tempCOM_V);

        // output regeneration level:  channel 2
        outportb (BASE_5710+14, tempREGEN % 256);
        tempREGEN = tempREGEN >> 8;
        outportb (BASE_5710+15, (unsigned char)tempREGEN);



        // output regeneration level:  channel 2
        // 0x00, 0x08 should produce 0V output
        //     0x00, 0x0F should produce 10V output
        // 0x00, 0x00 should produce -10V output

}
```

ZEV.H

```
// zev.h
// header file for the ZEV mode
#include <stdio.h>
#include <time.h>

class ZElectricV
{
        public:
                ZElectricV(void);
                ZElectricV(int){};
                ~ZElectricV(void);
                int EnterPoint(void);

/*      function prototypes */

        protected :
                float calcSOC(void);
                void getDriverInput();
                void OutputToMotor();
                void calcMotorOutput();
                void EverySec(void);


/*      variables */
/* for an explanation of the unit V*20, see main.cpp */

        protected:
                float curSOC;           // battery state of charge
                float curTPS;           // driver throttle position, %
                int curBPRESS;          // brake pressure, V*20
                float curREGEN; // regeneration level to motor, %
                float curCOM_V; // commanded speed to motor, %
                time_t preTime; // timer, s
                float InputTPS[50];     // array of curTPS; use in filter
                int ArrayCount; // counter for filter
};
```

HEV.CPP

```
// hev.cpp
#include <dos.h>
#include "hev.h"
#include "braking.h"
#include "assist.h"
#include "regen.h"
#include "main.h"
#include "apu.h"


#include <stdio.h>

/*
*       FILE:           hev.cpp
*
*       PURPOSE:         "hev.cpp", once called from "main.cpp" directs
*                       control of the Unique Mobility SR-180 & CR-20/300C
*                       to propel the vehicle in conjunction with the
*                       hybrid power unit (internal combustion engine).
*                        This involves choosing between four different
*                       operating states:  APU only, assist, regen, and
*                       braking.  These functions, once called, determine
*                       the required output to the motor and call the motor
*                       output function to execute these values.
*
*       PROCEDURE:      Step 1:  initialize classes and variables
*                       (Main Loop)
*                       Step 2:  monitor and display system state
*                       Step 3:  read data required for threshold calculation
*                       Step 4:  calculate thresholds
*                       Step 5:  determine operating state and call function
*
*
*       LINEAGE:        Brian E. Tucker, James Hatmaker         1.00   5/5/96
*                               - original version
*
*                       Brian E. Tucker                 1.01    11/9/96
*                               - added comment blocks
*
*                       Brian E. Tucker                 1.02    1/28/97
*                               - additional comments
*
*
*/


HElectricV::HElectricV()
{
}

HElectricV::~HElectricV()
{
}

int HElectricV::EnterPoint()
{

/*
*  STEP 1:  INITIALIZE CLASSES AND VARIABLES
*    This section involves the creation of instances of each of the four mode
classes
*   (brake, assist, regen, apu) as well as obtaining the settings from the
external
```

```
*   "settings.ini" file.  Finally, a few variables are defined.
*/

        /* define instances of mode classes */
        CBraking brake;
        CAssist assist;
        CRegen regen;
        APU apu;


        /* get settings imfo from file */
        SETTINGSSTRUCT* p_set = getSettings();

        preTime = time(0);
        int a = 0;

/***** MAIN LOOP *****/

        while (a < 1000)
        {
                a = a; // This ensure we never leave this loop (a la Hotel
California)

/*
*  STEP 2:  MONITOR AND DISPLAY SYSTEM STATE
*   Using the EverySec function, operations which have a lower priority than
*   powertrain control can be executed only during an arbitrary interval of
*   1 s.  These include safety monitoring, calculating battery SOC, and
*   updating driver display.
*/

                EverySec();     /* runs functions inside of function once a
second */


/*
*  STEP 3:  READ DATA REQUIRED FOR THRESHOLD CALCULATION
*    Here, the various parameters which are needed to determine the correct
thresholds
*     for the throttle position are read.   First gear assist is the only
currently active
*   value needed (variable = G1).  The other values related to the now defunct
*   driver-commanded (dicatator) assist and regeneration.
*/

                // read values to calculate thresholds
                // note:  these values use reverse logic !!!
                tmp = inportb(PORTC_5540);
                DICT_R = tmp & P2;              // dictator regen (defunct)
                DICT_A = tmp & P1;              // dictator assist (defunct)
                tmp = inportb(PORTA_5540);
                G1 = tmp & P7;                  // first gear assist

                //just for future use, print here to show time steps
                printf("  0\n ");

/*
*  STEP 4:  CALCULATE THRESHOLDS
*   Using the data gleaned from the previous step, this function determines the
*   appropriate high and low thresholds for the control value (here, TPS).
*   Therefore, its output is high TPS threshold and low TPS threshold.  These
*   values are used to determine which of the four modes should be active.
*/

                calcThresholds();

/*
```

```
*  STEP 5:  DETERMINE OPERATING STATE AND CALL FUNCTION
*      This final step cycles through each of the four operating modes
(regenerative
*     braking, assist, regeneration / charging, and apu-only).   Within each
function
*   call, it is determined whether the function should be in use.  If it passes
its
*   own internal "test," then the cycle of calling functions stops and this
*   function executes.
*/

        /* if the brake pressure level is greater than that set in the
           "settings.ini" file, then this function will execute its
           motor control routine.  Else, check next function. */

                if(brake.Brakemode(p_set))
                        continue;

        /* if the throttle position exceeds the high throttle threshold, then
           assist mode will be active.  Else, check next function. */

                if(assist.Assistmode(TPS_HI))
                        continue;

        /*  if  the  throttle  position  dips  below  the  low  throttle  position
threshold,
           then the regen mode will be active.  Else, check next function */

                if(regen.Regenmode(TPS_LO))
                        continue;


        /* if none of the above applies, the vehicle will default  to
           this setting - no motor interference, apu (HPU) only.*/

                apu.APUmode();
        }


        return 1;
}


void HElectricV::EverySec(void)
{

/*
*       EverySec(void)
*
*       PURPOSE:        Runs member functions once every second.  These
*                       functions are of lower priority than the power-
*                       train control functions and are arbitrarily
*                       executed only once a second.  These include
*                       reading data from serial port buffer, calculating
*                       battery state of charge, performing safety checks,
*                       and updating display.
*
*
*       LINEAGE:        Brian E. Tucker, James Hatmaker         1.00   5/5/96
*                               - original version
*
*                       Brian E. Tucker                 1.01   10/27/96
*                               - added comment blocks
*
*
*/
```

```
        time_t curtime = time(0);
        if((curtime - preTime) < 1)
                return;

        preTime = curtime;

 //     ReadSerial();                          //  reads  the  buffer  for  the
serial port
        // initial workings...
        curSOC = calcSOC();
        SAFETYSTRUCT safety = safetyCheck();
        Display();

        // data acq. 1-s warning
        printf("one second \n");
}


void HElectricV::calcThresholds(void)
{

/*
*       calcThresholds(void)
*
*       PURPOSE:        This function calculates the required thresholds for
*                       each of the classes based on secondary data.  Two0
*                       thresholds exist in each case:  TPS_HI and TPS_LO.
*                       If TPS_HI is exceeded, then assist will occur.
*                       If TPS falls below TPS_LO, regen will occur.  This
*                       function is where the values of the these thresholds
*                       are evaluated.
*
*       LINEAGE:        Brian E. Tucker, James Hatmaker         1.00   5/5/96
*                               - original version
*
*                       Brian E. Tucker                 1.01   4/9/97
*                               - added comment blocks
*
*
*/

        SETTINGSSTRUCT* set = getSettings();

        if (!G1)
        {
                /* if we're in first gear, then
                        high threshold = TPS_1G
                        low threshold  = 0. */

                TPS_HI = set->TPS_1G;
                TPS_LO = 0;
        }

        else if (!DICT_A)
        {
                /* if dictator assist enabled (now defunct)
                        high threshold = TPS_HIdicta
                        low threshold  = TPS_Lodicta */

                TPS_HI = set->TPS_HIdicta;
                TPS_LO = set->TPS_LOdicta;
        }

        else if (!DICT_R)
        {
                /* if dictator regen enabled (now charge-depleting)
```

```
                        high threshold = TPS_HIdictr
                        low threshold  = TPS_Lodictr */

                TPS_HI = set->TPS_HIdictr;
                TPS_LO = set->TPS_LOdictr;
        }

        else if (curSOC <= set->SOC_min)
        {
                /* if we've fallen below minimum SOC
                        high threshold = 100  (cut-off assist)
                        low threshold  = TPS_Lodictr */

                TPS_HI = 100;
                TPS_LO = set->TPS_LOdictr;
        }

        else if (curSOC > set->SOC_max)
        {
                /* if we've gone above maximum SOC
                        high threshold = TPS_HInorm
                        low threshold  = 0 (disable regen) */

                TPS_HI = set->TPS_HInorm;
                TPS_LO = 0;
        }

        else
        {
                /* default conditions */

                TPS_HI = set->TPS_HInorm;
                TPS_LO = set->TPS_LOnorm;
        }
        return;
}


void HElectricV::Display(void)
{
        return;
}


SAFETYSTRUCT HElectricV::safetyCheck(void)
{
        SAFETYSTRUCT safety;

        return safety;
}


float HElectricV::calcSOC()
{
/*
*       calcSOC(void)
*
*       PURPOSE:        Calculates state of charge (SOC) of the battery pack.
*                       The total used amp hours (read from KWHR meter) in the
*                       data structure are added (since this value is < 0) to
*                       the  battery  pack  capacity  as  entered  into  the
settings.ini
*                       file.  The resulting number (amp-hours of capacity
*                       available) is then divided by capacity to determine
*                       state of charge (in percent).
*
*
```

```
*       LINEAGE:        Brian E. Tucker, James Hatmaker         1.00   5/5/96
*                               - original version
*
*                       Brian E. Tucker                 1.01    1/28/97
*                               - added comment blocks
*
*
*/

        /* get "amp-hours used" (p_data->batAHRS) for KWHR meter */

        DATASTRUCT* p_data = getData();

        /* get "battery capacity" (p_set->batt_capacity) from settings.ini file
*/

        SETTINGSSTRUCT* p_set = getSettings();

        /* calculate state of charge */

        float   SOC   =   (p_data->batAHRS   +   p_set->batt_capacity)   /   p_set-
>batt_capacity * 100;

        return SOC;

}
```

HEV.H

```
// hev.h
// header file for the HEV mode

#include <time.h>

#ifndef m_hev
#define m_hev

typedef enum
{
        high_motor_temperature,
        high_controller_temperature,
        motor_not_ready,
        motor_fault,
        TPS_fault
} SAFETYSTRUCT;


class HElectricV
{
        public:
                HElectricV(void);
                HElectricV(int){};
                ~HElectricV(void);
                int EnterPoint();

protected:
        float calcSOC(void);
        SAFETYSTRUCT safetyCheck(void);
        void calcThresholds(void);
        void Display(void);
        void EverySec(void);
        void EveryHalfSec(void);
        int getRPM(void);

protected:
        int tmp;
        int G1;
        int DICT_R;
        int DICT_A;
        float curSOC;
        int TPS_HI;
        int TPS_LO;
        time_t preTime;
        time_t preHalfTime;
        int MotorRPM;


};

#endif
```

APU.CPP

```
// APU.cpp
#include <stdio.h>
#include "apu.h"
#include "motorout.h"

APU::APU(void)
{
}
APU::~APU(void)
{
}

void APU::APUmode(void)
{
//      printf("APU\n");
        curREGEN = 0.0;
        curCOM_V = 0.0;
        CMotorOutput motorOutput;
        motorOutput.HEVoutputToMotor(curREGEN,curCOM_V);
        return;
}
```

APU. H

```
// APU.h


#ifndef m_apu
#define m_apu

class APU
{
        public:
                APU(void);
                ~APU(void);
                void APUmode(void);


        protected:
                float curREGEN;
                float curCOM_V;

};

#endif
```

ASSIST.CPP

```
// assist.cpp
#include <stdio.h>
#include <dos.h>
#include "assist.h"
#include "motorout.h"

/*
*       FILE:           assist.cpp
*
*       PURPOSE:        the "assist.cpp" file is first called from the
*                       hev.cpp file.  It perform several A/D conversions
*                       to get values for V_act and motor current.
*                       The first step, however, checks to see if the
*                       assist TPS threshold has been exceeded.  If so,
*                       this file carries out assist.  If not, it returns.
*
*       PROCEDURE:      add this in later.
*
*
*       LINEAGE:        Brian E. Tucker, James Hatmaker         1.00   5/5/96
*                               - original version
*
*                       Brian E. Tucker                 1.10    2/16/97
*                               - added current feedback
*
*                       Brian E. Tucker                 1.11    4/11/97
*                               - added comment blocks
*
*/


CAssist::CAssist(void)
{
}
CAssist::~CAssist(void)
{
}

int CAssist::Assistmode(int tps_hi)
{

// current changes BET 2/16/97 - current feedback added (v1.10)
// future additions - add the ADconv. function.

        TPS_HI = tps_hi;
        // test if TPS > threshold
        if(!TestAssist())
                return 0;

        printf("ASSISTING");

        // calculate assist level here!!!

        // get actual speed

        outportb(CH_ADDR, CH_V_ACT);
        delay(1);
        outportb(BASE_5710, 0xFF);

        int tmp4 = inportb(BASE_5710);
        tmp4 &= 0x01;
        while(!tmp4)
        {
```

```
                tmp4 = inportb(BASE_5710);
                tmp4 &= 0x01;
        }

        // note: this value of V_ACT ranges from 2048-0 (i.e., 0 to -5 V )
        float curV_ACT = (inportb(BASE_5710+2)*16 + inportb(BASE_5710+3)/16);
        // convert to range of 0 - 100%
        curV_ACT = -2.0 * 100.0 / 2048.0 * (curV_ACT - 2048.0); // 0 < curV_ACT
< 100


        // get motor current


        outportb(CH_ADDR, CH_I_MOTOR);
        delay(1);
        outportb(BASE_5710, 0xFF);

        int tmp3 = inportb(BASE_5710);
        tmp3 &= 0x01;
        while(!tmp3)
        {
                tmp3 = inportb(BASE_5710);
                tmp3 &= 0x01;
        }

        // note: this value ranges from 2048-0 (i.e., 0 to -5 V )
        float curI_MOT = (inportb(BASE_5710+2)*16 + inportb(BASE_5710+3)/16);
        // convert to range of 0 - 100%
        curI_MOT = 1.0 * 100.0 / 2048.0 * (curI_MOT - 2048.0); // 0 < curI_MOT
< 100

        // now, convert % into amps.  If voltage divider cuts in in half, then
        // 2 V = 100 A (instead of 4 V in original 10 V signal.
        // Therefore, since 2 V = 40% = 100 A, multiply % by 2.5 to get A.
        // However, testing on 2/20/97 by BET showed that this value
        // was approximately twice as high as the KWHR meter output.
        // Therefore, divide by 2.

        curI_MOT *= 1.25;

        printf("curI_MOT = %f ",curI_MOT);


        float prevCOM_V = curCOM_V;


/*      Now, here's where the fun begins.  This is the new assist control
        block.  We're following this diagram:

        V_ACT ----------------------------------------------------|
                                                                  |
                  ____________                                    |
                 |            | I_des  (+)     I_err             \/ (+)
        curTPS-->|   G(s)     -------->( )----------| K_iv |----->(sum)-> V_com
                 |____________|       /\ (-)                  (+)
                                       |
        I_MOT ------------------------|

        Two feedback loops happen here.  First of all, the function G(s)
        relates TPS to desired motor current.  Subtract from this the
        actual motor current to determine error.  Multiply error by
        gain K_iv and add to V_ACT signal to get the required
        V_COM output signal.

*/
```

```
        // Assist Equation


        //linear case
        float I_des = (200.0/(90-TPS_HI)*(curTPS-TPS_HI));

        //hyperbolic case - added 2/21/97 by BET
//      float I_des = 288.0 - 7200.0 / curTPS;

        //standard; will not change
        float I_err = I_des - curI_MOT;


        // here, you're assuming K_iv value; make it a parameter!!
        // assume K_iv = 0.0025 V/A = 0.05 %/A

        float V_err = .10 * I_err;

        curCOM_V = V_err + curV_ACT;



        if(curCOM_V > 100.0)
                curCOM_V = 100.0;
        if (curCOM_V < 0.0)
                curCOM_V = 0.0;


        if((curCOM_V - prevCOM_V) > 5)
                curCOM_V = prevCOM_V + 5;



//      curCOM_V = 65;

        // set regen = 0
        curREGEN = 0.0;

        char tmpstr[100];
        sprintf(tmpstr,"curTPS = %f curCOM_V = %f curV_ACT = %f\n", curTPS,
curCOM_V, curV_ACT);
        printf(tmpstr);


        CMotorOutput motorOutput;
        motorOutput.HEVoutputToMotor(curREGEN,curCOM_V);


        return 1;
}

int CAssist::TestAssist(void)
{

        // lastest changes by BET, 2/16/97; change TPS scaling.

        // test if TPS is greater than TPS_HI.
        // return 1 if it is.

        // get TPS

        outportb(CH_ADDR, CH_TPS);
        delay(1);
        outportb(BASE_5710, 0xFF);

        int tmp = inportb(BASE_5710);
        tmp &= 0x01;
```

```
        while(!tmp)
        {
                tmp = inportb(BASE_5710);
                tmp &= 0x01;
        }

        // note: this value of TPS ranges from 2048-4096
        curTPS = (float)(inportb(BASE_5710+2)*16 + inportb(BASE_5710+3)/16);
        // convert to range of 0 - 100%
        curTPS = (float)((curTPS - 2048.0)/20.47); // 0 < curTPS < 100

        //What we really need now is to change this (TPS) to a true percent
        //basis.  0% TPS = 23, max TPS = 61 (based on past tuning)
        //these values came from ZEV.CPP dating 10/28/96

        curTPS = (float)(2.63157*(curTPS-23.0)); // true % TPS

        if (curTPS > TPS_HI)
                return 1;

        return 0;

}
```

ASSIST.H

```
// Assist.h

#include "main.h"

#ifndef m_assist
#define m_assist

class CAssist
{
        public:
                CAssist(void);
                CAssist(int){};
                ~CAssist(void);
                int Assistmode(int);
                int TestAssist(void);
//              float ADconv(int);

        protected:
                float curREGEN;
                float curCOM_V;
//              float curI_MOT;
                float curTPS;
                int TPS_HI;

};

#endif
```

BRAKING.CPP

```
// braking.cpp
#include <stdio.h>
#include <dos.h>
#include "hev.h"
#include "braking.h"
#include "motorout.h"

/*
*       FILE:           braking.cpp
*
*       PURPOSE:        The "braking.cpp" function follows the standard
*                       mode manager outline:  first, it checks to
*                       see if the 'brake pressure' signal exceeds the
*                       threshold value.  If so, the function sets the
*                       regen level and sends to the the motorout
*                       function.
*
*
*       PROCEDURE:      Step 1:  check thresholds
*                       Step 2:  calculate regen level
*                       Step 3:  call motorout function
*
*
*       LINEAGE:        Brian E. Tucker, James Hatmaker         1.00   5/5/96
*                               - original version
*
*                       Brian E. Tucker                 1.01    4/11/97
*                               - added comment blocks
*
*
*
*/


CBraking::CBraking(void)
{
}
CBraking::~CBraking(void)
{
}

int CBraking::Brakemode(SETTINGSSTRUCT* p_set)
{
        // test if braking
        if(!TestBrake(p_set))
                return 0;

//      printf("BRAKING\n");

        // get actual speed

        outportb(CH_ADDR, CH_V_ACT);
        delay(1);
        outportb(BASE_5710, 0xFF);

        int tmp4 = inportb(BASE_5710);
        tmp4 &= 0x01;
        while(!tmp4)
        {
                tmp4 = inportb(BASE_5710);
                tmp4 &= 0x01;
        }
```

```
        // note: this value of V_ACT ranges from 2048-0 (i.e., 0 to -5 V )
        float curV_ACT = (inportb(BASE_5710+2)*16 + inportb(BASE_5710+3)/16);
        // convert to range of 0 - 100%
        curV_ACT = -2.0 * 100.0 / 2048.0 * (curV_ACT - 2048.0); // 0 < curV_ACT
< 100


        int min_speed = 18;

        // calculate regen level here!!!

        if (curV_ACT > (float)min_speed)

                curREGEN = 90;
        else

                curREGEN = 0;

        // commanded speed = 0
        curCOM_V = 0.0;

        CMotorOutput motorOutput;
        motorOutput.HEVoutputToMotor(curREGEN,curCOM_V);
        return 1;
}

int CBraking::TestBrake(SETTINGSSTRUCT* p_set)
{
        // test if brake pressure is greater than min.
        // return 1 if braking

        outportb(CH_ADDR, CH_BPRESS);

        delay(1);

        outportb(BASE_5710, 0xFF);

        int tmp3 = inportb(BASE_5710);
        tmp3 &= 0x01;
        while(!tmp3)
        {
                tmp3 = inportb(BASE_5710);
                tmp3 &= 0x01;
        }


        int highBrake = inportb(BASE_5710+2);
        int lowBrake = inportb(BASE_5710+3);


        // note: this value of BPRESS ranges from 2048~4096
        curBPRESS = (int)(highBrake*16 + lowBrake/16);
        // convert to range of 0 - 100%
        curBPRESS = (int)((curBPRESS - 2048)/20.47); // 0 < BPRESS < 100

//      char tmpstr[100];
//      sprintf(tmpstr,"curBPRESS = %f highBrake = %d\n",curBPRESS,highBrake);
//      printf(tmpstr);

        if (curBPRESS > p_set->BPRESS_min)
                return 1;

        return 0;

}
```

BRAKING.H

```
// Braking.h

#include "main.h"

#ifndef m_braking
#define m_braking

class CBraking
{
        public:
                CBraking(void);
                CBraking(int){};
                ~CBraking(void);
                int Brakemode(SETTINGSSTRUCT*);
                int TestBrake(SETTINGSSTRUCT*);

        protected:
                float curREGEN;
                float curCOM_V;
                float curBPRESS;

};

#endif
```

REGEN.CPP

```
// regen.cpp
#include <dos.h>
#include <stdio.h>
#include "regen.h"
#include "motorout.h"


/*
*       FILE:           regen.cpp
*
*       PURPOSE:        "regen.cpp" continues the role of manager by first
*                       checking to see if the regen threshold has been
*                       met.  If so, it calculates the regen level and
*                       sends that to the motorout function.
*
*       PROCEDURE:      Step 1:  check threshold criteria
*                       Step 2:  calculate regen level
*                       Step 3:  output levels to motorout function
*
*       LINEAGE:        Brian E. Tucker, James Hatmaker         1.00   5/5/96
*                               - original version
*
*                       Brian E. Tucker                 1.10    10/26/96
*                               - changed TPS scaling
*
*                       Brian E. Tucker                 1.11    4/11/97
*                               - added comment blocks
*
*
*/

CRegen::CRegen(void)
{
}
CRegen::~CRegen(void)
{
}

int CRegen::Regenmode(int tps_lo)
{
        TPS_LO = tps_lo;
        // test if braking
        if(!TestRegen())
                return 0;

        printf("REGEN\n");

        // get actual speed

        outportb(CH_ADDR, CH_V_ACT);
        delay(1);
        outportb(BASE_5710, 0xFF);

        int tmp4 = inportb(BASE_5710);
        tmp4 &= 0x01;
        while(!tmp4)
        {
                tmp4 = inportb(BASE_5710);
                tmp4 &= 0x01;
        }

        // note: this value of V_ACT ranges from 2048-0 (i.e., 0 to -5 V )
        float curV_ACT = (inportb(BASE_5710+2)*16 + inportb(BASE_5710+3)/16);
```

```
        // convert to range of 0 - 100%
        curV_ACT = -2.0 * 100.0 / 2048.0 * (curV_ACT - 2048.0); // 0 < curV_ACT
< 100



        // calculate regen level here!!!

        int REGEN_MAX = 60;
        int min_speed = 18;

        if (curV_ACT > (float)min_speed)

        {

                // linear scheme for regen level:  0 - 100 scale
                curREGEN = (float)(-1 * (float)REGEN_MAX /((float)(TPS_LO+1))) *
(curTPS -(float)TPS_LO);
//              curREGEN = 50.0/20.0 * (20.0 - curTPS);
        }

        else

                curREGEN = 0.0;

//      printf("curREGEN = %f\n",curREGEN);
        // set COM_V = 0
        curCOM_V = 0.0;


        CMotorOutput motorOutput;
        motorOutput.HEVoutputToMotor(curREGEN,curCOM_V);
        return 1;
}

int CRegen::TestRegen(void)
{
        // test if TPS is less than TPS_LO.
        // return 1 if it is.

        // get TPS

        outportb(CH_ADDR, CH_TPS);
        delay(1);
        outportb(BASE_5710, 0xFF);

        int tmp3 = inportb(BASE_5710);
        tmp3 &= 0x01;
        while(!tmp3)
        {
                tmp3 = inportb(BASE_5710);
                tmp3 &= 0x01;
        }

        int tpsHigh = inportb(BASE_5710+2);
        int tpsLow  = inportb(BASE_5710+3);


        // note: this value of TPS ranges from 2048-4096
        curTPS = (tpsHigh*16 + tpsLow/16);
        // convert to range of 0 - 100%
        curTPS = (float)((curTPS - 2048)/20.47); // 0 < curTPS < 100

        //What we really need now is to change this (TPS) to a true percent
        //basis.  0% TPS = 23, max TPS = 61 (based on past tuning)
        //these values came from ZEV.CPP dating 10/28/96
```

```
        curTPS = (float)(2.63157*(curTPS-23.0)); // true % TPS


        char tmpstr[100];
        sprintf(tmpstr, "curTPS = %d TPS_LO = %d",curTPS, TPS_LO);
        printf(tmpstr);

        if (curTPS < (float)TPS_LO)
                return 1;

        return 0;
}
```

REGEN. H

```
// Regen.h

#include "main.h"

#ifndef m_regen
#define m_regen

class CRegen
{
        public:
                CRegen(void);
                CRegen(int){};
                ~CRegen(void);
                int Regenmode(int);
                int TestRegen(void);

        protected:
                float curREGEN;
                float curCOM_V;
                float curTPS;
                int TPS_LO;

};

#endif
```

MOTOROUT.CPP

```
#include <dos.h>
#include <stdio.h>
#include "main.h"
#include "motorout.h"

/*
*       FILE:           motorout.cpp
*
*       PURPOSE:        Sweet and simple, "motorout.cpp" receives values
*                       of regen level and commanded speed and then
*                       converts them into the scales needed for the
*                       D/A converter and thus the Unique.
*
*       PROCEDURE:      N-A
*
*
*       LINEAGE:        Brian E. Tucker, James Hatmaker         1.00   5/5/96
*                               - original version
*
*                       Brian E. Tucker                 1.01    4/11/97
*                               - added comment blocks
*
*
*
*/


CMotorOutput::CMotorOutput(void)
{
}

CMotorOutput::~CMotorOutput(void)
{
}

int CMotorOutput::HEVoutputToMotor(float curREGEN, float curCOM_V)
{
                // convert regen, com_v from 0-100 to 0 - 4095
         int COM_V = (int)curCOM_V * 20.47 + 2048;

         // just in case it might go negative...it won't!
         if (COM_V <= 2048) COM_V=2048;

         // check regen on/off switch
         int tmp = inportb(PORTA_5540);
         tmp &= P2;

         // if regen off, don't do it man!!!  Otherwise, it's cooool.
         int REGEN;
         if (!tmp)
         {
               REGEN = (int)(-1.0 * curREGEN)* 20.47 + 2048;
//             printf("regen on\n");
         }
         else
         {
               REGEN = 2048;
//             printf("               regen off\n");
         }

         //convert values to two registers needed for analog output
         int tempCOM_V = COM_V;
         int tempREGEN = REGEN;
```

```
        // output commanded speed:  channel 1
        outportb (BASE_5710+12, (unsigned char)(tempCOM_V));
        tempCOM_V = tempCOM_V >> 8;
        outportb (BASE_5710+13, (unsigned char)tempCOM_V);

        // output regeneration level:  channel 2
        outportb (BASE_5710+14, (unsigned char)tempREGEN);
        tempREGEN = tempREGEN >> 8;
        outportb (BASE_5710+15, (unsigned char)tempREGEN);
       return 1;
}
```

MOTOROUT.H

```
// motorout.h
// header file for the

#ifndef m_motorout
#define m_motorout

class CMotorOutput
{
public:
        CMotorOutput();
        ~CMotorOutput();
        int HEVoutputToMotor(float, float);
};

#endif
```

SETTINGS.INI

Note: this file is used to set certain control constants. It simply consists of the numerical values of each of these constants; the explanations of each are added here for convenience.

17.0 :	battery capacity, amp-hours (batt_capacity)
15.0 :	minimum brake pressure (BPRESS_min)
80.0 :	maximum brake pressure (BPRESS_max)
22.0 :	minimum TPS (TPS_min)
70.0 :	maximum TPS (TPS_max)
60 :	minimum battery SOC (SOC_min)
95 :	maximum battery SOC (SOC_max)
25 :	default high TPS threshold (TPS_HInorm)
20 :	default low TPS threshold (TPS_LOnorm)
20 :	dictator assist high TPS threshold (TPS_HIdicta)
0 :	dictator assist low TPS threshold (TPS_LOdicta)
25 :	dictator regen high TPS threshold (TPS_HIdictr)
0 :	dictator regen low TPS threshold (TPS_LOdictr)
5 :	first gear assist high TPS threshold (TPS_1G)
10 :	TPS filter parameter (AVG_CYCLE)

Appendix D

TESTING EQUIPMENT

Chassis Dynamometer

Manufacturer:	Sun Electric (Chicago IL)
Model:	Road - A - Matic RAM-937-1
Serial Number:	74D-0046

Data Acquisition System

Manufacturer:	Cruising Equipment
Model:	KWHR 2+
Serial Number:	110

Variables measured:

battery voltage, battery current, CNG pressure, CNG temperature

Variables calculated:

battery capacity used

Battery Voltage

Range:	0-500 V
Least Count:	0.1 V

Battery Current

Method:	measurement of voltage drop across shunt
Range:	0-500 A
Least Count:	0.1 A

CNG Pressure

Method:	pressure sensor (per spec below)
Manufacturer:	Omega
Model:	PX176
Range:	0-5000 psi
Accuracy:	1 % FS (50 psi)

CNG Pressure

Method:	thermistor

Appendix E

TEST RESULTS

Tests are broken down into the following categories.

- 40-50 mph acceleration tests
- 30-20 mph deceleration tests
- First gear assist tests
- braking tests

These will each be presented in detail through tabulated numerical results.

40-50 acceleration test

cycle: 4050a.cyc

		without assist	with assist						
			assist 1		assist 2			assist 3	
time	**Desired Speed**	**4050wo a2**	**4050 wa1**	**4050 wa2**	**4050 wa3**	**4050 wa4**	**4050 wa5**	**4050 wa6**	**4050 wa7**
0	40	38.7	39.5	39.0	40.3	40.1	39.1	40.1	38.7
0.5	40	39.3	39.5	39.1	40.4	40.6	39.4	40.3	39.2
1	40	39.5	39.8	39.3	40.7	40.7	39.8	40.6	39.6
1.5	40	39.8	40.3	39.0	41.3	41.1	39.8	40.9	39.5
2	40	40.2	40.5	39.1	41.2	41.2	40.1	41.0	40.0
2.5	40	40.5	40.8	38.8	41.7	41.7	40.3	41.0	40.2
3	40	40.9	41.0	39.1	41.9	41.6	40.4	41.4	40.5
3.5	41	41.2	41.3	39.2	42.2	42.2	41.2	41.4	40.9
4	42	41.6	41.7	39.6	42.5	42.6	42.0	41.8	42.7
4.5	43	42.0	42.1	41.9	43.1	43.4	42.9	42.3	44.4
5	44	42.3	42.6	43.5	43.7	44.4	43.6	44.6	45.7
5.5	45	42.7	42.9	44.8	43.9	45.2	44.5	46.2	47.3
6	46	43.0	43.3	45.4	44.7	46.0	45.4	47.6	48.4
6.5	47	43.4	43.7	45.8	45.0	46.5	46.0	49.1	49.7
7	48	43.8	44.0	46.5	45.3	47.0	46.9	50.0	51.0
7.5	49	44.1	44.4	48.1	46.2	47.7	47.7	50.8	51.9
8	50	44.3	44.8	49.5	46.9	48.1	48.3	51.9	51.4
8.5	50	44.8	45.1	50.3	46.0	48.4	49.0	50.7	51.4
9	50	45.2	45.3	50.5	44.1	49.1	50.0	49.4	51.4
9.5	50	45.4	45.8	49.3	41.9	49.8	50.5	48.6	51.2
10	50	45.8	46.1	47.1	39.4	50.2	51.1	48.1	51.5
10.5	50	46.3	46.5	44.2	36.6	51.1	51.3	48.0	51.6
11	50	46.4	46.8	41.3	33.7	49.1	50.0	47.7	51.8

40-50 acceleration test

file: 4050wa4.daq

time	motor current/2	TPS	COM_V	V_ACT
3.5	1.7	43.3	40.2	38.6
3.6	0.7	99.1	45.2	38.7
3.7	4.2	99.0	50.2	39.0
3.8	51.4	99.0	55.2	40.1
3.9	73.0	99.7	60.2	45.2
4	75.6	97.6	62.1	47.4
4.1	70.6	99.7	67.0	51.1
4.2	62.7	98.2	70.2	53.9
4.3	66.2	99.4	72.6	56.3
4.4	59.6	102.6	76.6	58.7
4.5	56.9	101.5	78.2	60.4
4.6	53.6	100.2	79.4	61.6
4.7	52.6	99.3	80.0	62.4
4.8	54.9	99.8	80.1	62.6
4.9	49.7	92.7	79.3	63.5
5	48.3	88.2	78.9	64.3
5.1	46.5	91.4	80.1	64.4
5.2	48.8	91.3	80.8	65.2
5.3	50.7	88.9	80.1	65.5
5.4	42.7	87.7	80.7	65.7
5.5	47.3	95.0	82.7	65.9
5.6	40.5	90.4	82.0	65.9
5.7	45.4	93.4	82.5	66.0
5.8	47.1	100.3	84.8	66.3
5.9	43.3	100.8	85.2	66.2
6	46.0	100.6	85.1	66.4
6.1	42.6	100.7	85.8	66.8
6.2	41.9	99.5	85.0	66.3
6.3	40.2	101.2	85.9	66.5
6.4	38.8	102.2	86.5	66.6
6.5	39.4	101.3	87.4	67.9
6.6	42.1	101.1	86.6	67.4
6.7	39.2	103.8	88.0	67.7
6.8	38.8	100.9	87.1	67.6
6.9	34.7	102.5	88.7	68.4
7	36.4	102.4	88.8	68.7
7.1	37.0	104.4	89.5	68.8
7.2	36.6	110.6	91.4	68.8
7.3	36.6	105.3	90.0	68.9
7.4	34.6	101.1	89.4	69.4

Continued...

time	motor current/2	TPS	COM_V	V_ACT
7.5	35.0	103.1	89.9	69.3
7.6	31.9	100.7	89.6	69.5
7.7	36.3	102.4	89.5	69.3
7.8	36.1	102.7	89.2	68.8
7.9	36.1	100.6	88.9	69.2
8	33.7	100.9	89.6	69.6
8.1	34.8	99.4	88.6	69.1
8.2	34.7	101.2	89.9	69.9
8.3	34.2	101.3	91.1	71.0
8.4	33.3	102.7	90.1	69.5
8.5	35.7	103.6	90.2	69.5
8.6	32.7	104.3	90.8	69.6
8.7	35.0	101.2	88.9	68.9
8.8	35.5	104.8	90.2	69.2
8.9	33.8	106.0	90.5	68.9
9	40.0	105.2	89.6	68.9
9.1	34.0	106.9	90.5	68.8
9.2	41.1	102.4	88.4	68.8
9.3	36.2	102.2	88.8	68.7
9.4	35.9	104.8	89.6	68.7
9.5	36.1	98.5	87.5	68.5
9.6	35.9	61.6	76.3	68.7

40-50 acceleration test

4050wa5.daq

time	motor current/2	TPS	COM_V	V_ACT
3	1.8	25.2	38.8	38.9
3.1	0.7	102.1	43.8	37.4
3.2	0.6	98.9	48.8	37.3
3.3	54.3	100.3	53.8	37.3
3.4	68.1	98.8	57.7	41.8
3.5	84.4	99.1	59.6	45.2
3.6	57.3	99.8	64.6	49.8
3.7	58.0	100.9	69.6	52.3
3.8	58.5	100.8	72.6	55.1
3.9	64.3	103.4	75.8	58.1
4	53.3	100.8	78.4	60.4
4.1	50.2	100.0	79.6	61.5
4.2	51.0	85.8	76.6	63.0
4.3	48.3	81.9	76.2	63.5
4.4	47.7	86.5	78.3	64.2
4.5	46.8	85.9	78.8	64.7
4.6	45.1	86.2	79.0	64.6
4.7	44.7	87.7	79.4	64.6
4.8	46.0	83.8	78.0	64.5
4.9	44.3	83.1	78.1	64.6
5	45.3	87.3	79.3	64.6
5.1	42.6	102.0	83.8	64.4
5.2	45.0	100.3	83.3	64.6
5.3	45.0	100.0	83.1	64.6
5.4	44.8	100.0	82.9	64.3
5.5	46.5	100.3	82.6	64.1
5.6	45.2	99.5	82.2	63.8
5.7	44.5	98.2	81.8	63.7
5.8	47.2	101.3	82.5	63.8
5.9	44.7	100.3	82.5	63.8
6	45.2	106.1	83.9	63.5
6.1	44.1	100.3	82.5	63.8
6.2	45.3	103.0	82.9	63.5
6.3	47.0	101.8	82.4	63.5
6.4	47.2	103.9	83.4	63.9
6.5	44.7	103.8	83.2	63.5
6.6	46.3	98.9	81.4	63.3
6.7	45.9	98.4	80.8	62.8

Continued

time	motor current/2	TPS	COM_V	V_ACT
6.8	46.8	99.0	81.4	63.3
6.9	45.3	99.8	81.9	63.4
7	48.8	99.1	80.9	63.0
7.1	47.1	101.2	81.7	63.0
7.2	44.9	98.9	81.1	62.9
7.3	45.0	97.7	79.7	61.8
7.4	47.1	97.7	80.6	62.9
7.5	44.8	100.7	81.7	62.9
7.6	48.9	101.1	81.2	62.7
7.7	46.6	99.1	80.9	62.7
7.8	44.5	98.6	81.3	63.1
7.9	46.8	98.8	80.5	62.5
8	45.5	100.0	81.0	62.5
8.1	46.3	97.2	80.0	62.4
8.2	46.1	96.6	80.8	63.4
8.3	48.3	97.3	79.2	61.8
8.4	44.4	98.0	80.2	62.2
8.5	42.0	97.5	80.1	62.0
8.6	48.2	98.4	79.9	62.1
8.7	46.3	99.8	80.4	62.0
8.8	46.0	97.1	79.8	62.2
8.9	43.9	97.1	79.9	62.1
9	44.4	97.0	79.9	62.2
9.1	45.7	97.5	79.8	62.1
9.2	45.9	96.7	79.6	62.1
9.3	42.1	96.2	79.8	62.1
9.4	47.7	99.8	80.5	62.3
9.5	44.6	88.2	77.6	62.6

40-50 acceleration test

file: 4050wa6.daq

time	motor current/2	TPS	COM_V	V_ACT
3.6	1.6	26.0	5.0	38.2
3.7	0.8	52.0	10.0	38.3
3.8	2.1	56.2	15.0	38.3
3.9	1.6	56.3	20.0	37.0
4	1.6	52.1	25.0	38.8
4.1	1.9	45.7	30.0	38.5
4.2	1.6	43.2	35.0	38.0
4.3	1.6	43.5	40.0	38.7
4.4	0.4	61.1	45.0	39.1
4.5	2.0	98.1	50.0	38.9
4.6	55.2	101.6	55.0	39.6
4.7	101.9	101.1	50.2	38.7
4.8	103.1	98.9	51.0	39.8
4.9	78.8	101.5	54.2	40.3
5	87.0	103.9	53.6	40.4
5.1	102.8	102.6	52.6	41.1
5.2	89.7	102.6	52.7	39.8
5.3	81.2	100.7	55.2	41.7
5.4	102.7	97.9	53.5	42.3
5.5	92.0	98.8	54.8	42.5
5.6	83.7	97.2	55.8	42.8
5.7	88.6	99.4	56.6	43.9
5.8	86.1	97.6	56.5	43.7
5.9	94.1	99.4	55.6	43.5
6	85.1	96.8	56.5	43.7
6.1	78.1	96.8	57.7	44.1
6.2	89.5	96.6	56.9	44.5
6.3	84.1	93.9	57.3	44.6
6.4	80.0	88.7	57.7	45.0
6.5	92.5	77.4	54.3	44.0
6.6	73.4	77.9	57.3	45.1
6.7	75.2	81.0	56.9	44.5
6.8	74.9	76.9	57.8	45.9
6.9	72.9	79.7	57.6	45.1
7	74.6	77.4	58.0	46.0
7.1	83.9	73.6	56.1	45.5
7.2	79.1	76.0	58.0	46.6
7.3	64.5	80.2	60.3	46.9

Continued

time	motor current/2	TPS	COM_V	V_ACT
7.4	82.5	79.7	58.5	47.0
7.5	86.5	69.8	57.0	47.2
7.6	68.8	64.6	58.0	47.3
7.7	67.9	43.5	52.2	46.8

40-50 acceleration test

file: 4050wa7.daq

time	motor current/2	TPS	COM_V	V_ACT
3.8	2.0	27.7	39.8	37.2
3.9	0.7	82.6	44.8	37.5
4	2.2	97.7	49.8	38.5
4.1	64.4	96.8	53.1	38.2
4.2	100.6	98.9	50.0	38.6
4.3	99.2	99.0	50.4	38.8
4.4	92.9	99.3	51.5	39.3
4.5	91.3	99.0	52.0	39.6
4.6	95.2	98.2	51.9	39.9
4.7	83.4	97.6	52.0	38.9
4.8	84.7	97.2	53.4	40.4
4.9	81.2	99.7	53.8	40.3
5	89.5	98.0	52.9	40.4
5.1	82.2	98.2	54.9	41.6
5.2	73.2	99.0	55.9	41.7
5.3	100.5	98.2	53.5	42.1
5.4	93.8	99.9	54.5	42.3
5.5	88.4	101.2	55.7	42.9
5.6	86.5	99.4	55.6	42.7
5.7	91.7	100.3	55.7	43.3
5.8	82.9	100.8	56.6	43.3
5.9	97.8	101.6	55.6	43.7
6	74.0	99.5	58.2	44.0
6.1	95.1	102.7	56.7	44.4
6.2	90.1	103.1	56.8	44.0
6.3	80.0	102.2	58.5	44.7
6.4	85.5	99.8	57.6	44.5
6.5	84.5	98.8	58.4	45.3
6.6	87.3	99.5	57.8	44.9
6.7	80.6	99.1	59.3	45.8
6.8	88.0	100.6	58.6	45.8
6.9	94.8	100.2	57.6	45.5
7	77.3	99.8	59.5	45.6
7.1	75.1	98.5	60.2	46.2

Continued

time	motor current/2	TPS	COM_V	V_ACT
7.2	91.3	100.7	59.4	46.9
7.3	84.3	99.5	59.9	46.8
7.4	78.3	102.4	61.0	47.1
7.5	84.0	83.8	59.7	47.9
7.6	81.0	74.7	58.5	47.5
7.7	77.3	34.1	47.4	47.5

For all acceleration test data, speed is expressed in mph (miles/hour). Time is in seconds, motor current is in Amps, TPS (throttle position) is in % (values above 100% demonstrate the calibration technique which allows 100% throttle at a value lower than the maximum possible throttle angle), and the COM_V and V_ACT signals are both expressed in % (of the input/output voltage signals).

30-20 deceleration test

cycle: 3020a.cyc

		without regen	with regen regen 1		
time	**Desired Speed**	**3020wor1**	**3020wr1**	**3020wr2**	**3020wr3**
0	30	29.3	30.3	30.3	30.9
0.5	30	30.1	30.4	30.4	31.1
1	30	30.4	30.5	30.5	31.3
1.5	30	30.4	30.4	30.4	31.8
2	30	31.1	30.7	30.7	31.4
2.5	30	31.2	30.7	30.7	31.2
3	30	31.6	30.7	30.7	30.7
3.5	20	31.8	30.5	30.5	30.2
4	20	31.2	29.5	29.5	29.3
4.5	20	30.4	28.5	28.5	28.5
5	20	29.9	27.4	27.4	27.8
5.5	20	29.7	26.7	26.7	26.7
6	20	29.3	25.8	25.8	26.1
6.5	20	29.2	24.9	24.9	25.2
7	20	29.1	24.5	24.5	24.4
7.5	20	28.8	23.7	23.7	23.9
8	20	28.4	22.9	22.9	23.4
8.5	20	28.4	22.5	22.5	22.6
9	20	28.4	22.1	22.1	22.0
9.5	20	28.0	21.5	21.5	21.7
10	20	27.9	20.9	20.9	21.2
10.5	20	27.6	20.5	20.5	20.7
11	20	27.5	19.9	19.9	20.4

First Gear Assist - Test 1
cycle:
1gramp.cyc

		without assist			with assist		
time	**Desired Speed**	**1grwoa1**	**1grwoa2**	**1grwoa3**	**1grwa1**	**1grwa2**	**1grwa3**
0	0	0.2	0.2	0.1	1.8	1.7	1.7
0.5	0	0.2	0.4	0.2	2.1	1.4	1.7
1	0	0.2	0.3	0.1	1.5	1.2	1.8
1.5	0	0.2	0.1	0.3	2.7	0.5	0.3
2	0	0.4	0.3	0.1	2.7	0.2	1.0
2.5	0	0.1	0.2	0.4	2.6	1.6	1.9
3	1	0.3	0.2	0.5	0.3	1.5	1.9
3.5	3	0.7	2.1	3.3	1.6	2.4	1.0
4	5	2.9	5.5	7.0	5.9	6.8	4.2
4.5	7	6.4	8.9	8.6	7.7	10.7	8.4
5	9	10.4	10.1	9.9	8.7	13.9	11.6
5.5	11	13.2	11.7	11.1	9.8	15.1	14.6
6	13	14.3	12.9	12.6	11.3	16.3	15.5
6.5	15	15.7	14.5	13.9	12.9	17.4	16.7
7	16	16.7	16.1	15.3	14.5	18.6	17.7
7.5	17	17.6	17.3	16.9	15.9	19.1	17.9
8	16	17.5	17.6	18.1	17.3	17.0	15.8
8.5	16	16.3	15.6	16.3	18.4	15.2	15.0
9	15	14.7	13.8	14.5	17.5	14.1	14.4
9.5	14	14.0	13.0	13.8	14.9	13.8	14.1
10	14	13.8	14.0	13.9	13.7	13.3	14.0
10.5	14	13.1	15.1	13.8	14.1	13.2	13.5
11	14	13.1	15.8	13.4	15.1	13.0	13.4

40-20 braking test

cycle: 4020a.cyc

		without regen	with regen regen 1
time	**Desired Speed**	**4020wor1**	**4020wr1**
0	40	39.7	38.9
0.5	40	40.3	39.1
1	40	40.6	39.5
1.5	40	40.8	39.3
2	40	41.1	39.7
2.5	40	41.3	39.7
3	40	41.6	39.8
3.5	20	41.8	36.7
4	20	37.4	29.9
4.5	20	28.9	22.0
5	20	20.5	14.1
5.5	20	12.1	5.3
6	20	1.9	0.7
6.5	20	0.7	0.6
7	20	0.6	0.7
7.5	20	0.3	0.6
8	20	0.5	0.4
8.5	20	0.5	0.6
9	20	0.7	0.7
9.5	20	0.6	0.6
10	20	0.7	0.5
10.5	20	0.7	0.7
11	20	0.5	0.4

Energy Efficiency Test
cycle:
FUD1.cyc

		Charge-sustaining			Charge-depleting			
					with regen. braking		without regen. braking	
test		1	2	4	6	7	3	5
date		11-Mar	12-Mar	17-Mar	18-Mar	18-Mar	12-Mar	17-Mar
Battery								
starting KWHR		0	-0.01	0.12	0.25	1.6	0	0.13
ending KWHR		-0.15	-0.05	0.14	0.09	1.42	-0.17	0
starting voltage		198	194	195.5	196	197	197	193.5
ending voltage		196	196	194.5	189	195.5	194.5	187.5
Fuel								
starting press. (psi)		2030	1884	1580	1225	1150	1840	1420
starting temp. (C)		20	19	17	17	19	21	16
ending press. (psi)		1870	1770	1410	1150	985	1674	1245
ending temp. (C)		19	17	16	19	18	20	15
V/V-start		174.5	162.1	135.1	101.2	93	155.9	120.5
V/V-end		160.8	153.4	119.5	93	78.5	141.4	104.3
LHV	Btu/SCF	927.9	927.9	927.9	927.9	927.9	927.9	927.9
delVstd	SCF	36.5242	23.1942	41.5896	21.8612	38.657	38.657	43.1892
LHV*Vstd	Btu	33890.81	21521.9	38590.99	20285.01	35869.83	35869.83	40075.26
gas equiv.	gal	0.296015	0.187981	0.337069	0.177177	0.313301	0.313301	0.350033
mileage	mpg	25.33652	39.89774	22.25066	42.33052	23.93864	23.93864	21.42656
elec. energy	kWhr	0.15	0.04	-0.02	0.16	0.18	0.17	0.13

Fuel Economy Calculations

V/V values represent relationship between volume of CNG at tank pressure to volume of CNG in equivalent standard ft^3 (SCF). The following assumptions apply:

1. barometric pressure = 25 in-Hg

2. tank volume = 75495 cm^3 = 2.666 ft^3

3. energy in gasoline = 114490 Btu/gal

4. total length of FUDS cycle = 7.5 miles

5. natural gas composition per [34]

The V/V values were generated using the computer program as referenced in [35].

Sample Calculation - Fuel Economy (mpg-equivalent)

$$V(SCF) = \left(V/V_{start} - V/V_{end}\right)V_{tank}$$

$$Energy = LHV\left(V(SCF)\right)$$

$$mpge = \left\{\left(\frac{Energ(Btu)}{114490 Btu/gal}\right)\frac{1}{7.5 miles}\right\}^{-1}$$

Uncertainty Analysis

As stated in Appendix D, the accuracy of the CNG pressure measurement is limited to 1% of full scale of the transducer which amounts to 50 psi. Taking the average starting pressure of 1590 psi and assuming the actual starting value could be as much as 50 psi lower (it could also be 50 psi higher), the following analysis has been made:

V/V-start		131.1
V/V-end		126.5
LHV	Btu/SCF	927.9
delVstd	SCF	12.2636

LHV*Vstd	Btu	11379.39
gas equiv.	gal	0.099392

This final value of equivalent gasoline consumption indicates the possible error in the calculated fuel consumption. For the lowest value of fuel used (0.177177 gal equivalent in Test 6), this amounts to a 56.1% error; for the highest amount of fuel used in a test (0.350033 gal equivalent in Test 5), the error is still significant at 28.4%. Therefore, values of fuel consumption could range from approximately 21-63 mpge equivalent (±58% at 42 mpge in Test 6). No conclusions can thus be made from the fuel economy data due to lack of accuracy in the pressure measurement device.

VITA

Brian E. Tucker was born on May 17, 1973 in Maryville, Tennessee as his parents' first child. He has two younger brothers—Kevin and David. He grew up in the same house and attended public schools in Blount County until eigth grade when his father's new job necessitated a move to Florence, Alabama. Brian always cherishes those early years and admits the most significant event of his life occurred there in 1981 when he understood for the first time how much he needed God's forgiveness and that God would freely give to him a new life. He has never been the same since.

Through the move to Alabama, Brian developed a strong interest in music which drove him to begin piano lessons. This love of music has always since been a source of joy and peace from God. The year 1988 saw another location change; the Tucker family moved to Cleveland, Tennessee where Brian attended and graduated as one of three valedictorians from Cleveland High School. During high school, Brian was heavily involved in the academic bowl team; this involvement afforded the opportunity to participate in three national tournaments. He also enjoyed drama, working on the staff of the literary magazine, and became active at North Cleveland Baptist Church.

Brian began work on a bachelor's degree in mechanical engineering in the fall of 1991 at the University of Tennessee. He was awarded an Andrew D. Holt Scholarship as well as a Fred M. Roddy Foundation scholarship and was a Chancellor's Scholar. All during college, he was active at the University of Tennessee Baptist Student Union (serving as vice president his senior year) and Calvary Baptist Church. God did many things during this time to bring Brian closer to himself during these years. Brian finished his undergraduate degree in

the spring of 1995 and began graduate studies at UT in the fall. Brian was awarded a Hilton A. Smith graduate fellowship to support his studies. During that summer, God gave Brian his second greatest gift: a friendship with the wonderful woman who would become his wife, Sherrie Ann Walker. Brian and Sherrie were wed on June 1, 1996. Upon completion of his master's degree in mechanical engineering in the spring of 1997, Brian plans to serve God with his wife in the Dallas-Ft. Worth area as he begins his professional career with Bell Helicopter. Brian looks forward to all that God will continue to do in the coming years; he can't believe its been this wonderful so far. Brian gives God all the glory for anything that he has been able to accomplish.

Matthew 6:33 "Seek first His kingdom and His righteousness and all these things will be added to you as well."